

# PATENT ABSTRACTS OF JAPAN

(11)Publication number : 11-272847

(43)Date of publication of application : 08.10.1999

(51)Int.Cl.

G06T 1/00

H04N 11/04

(21)Application number : 10-074719 (71)Applicant : SEIKO EPSON CORP

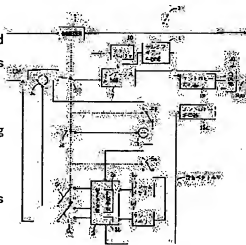
(22)Date of filing : 23.03.1998 (72)Inventor : HOSHINA SHOJI

(54) DEVICE AND METHOD FOR ENCODING MOVING IMAGE CONSISTING OF MULTICOLOR IMAGE, AND DEVICE AND METHOD FOR DECODING THE SAME MOVING IMAGE

(57)Abstract:

PROBLEM TO BE SOLVED: To apply motion compensation even when a moving image consisting of multicolor images is encoded and decoded and to attain the conflicting purposes of making a strict identification of a block at the time of motion compensation and of preventing an increase in the number of moving vectors.

SOLUTION: An encoding device 1 for moving image consisting of multicolor images performs motion compensation when generating a moving image. A judging means 2a calculates the number of different pixels at corresponding positions of both blocks and judges that the blocks are the same or similar when the number of different pixels is less than a specific value to generate a moving vector V. The judging means 2a also judges whether or not the moving vector V is used by divided block, and judges that the



blocks are the same or similar when the number of differences between both the blocks is less than a specified number to perform motion compensation by using the moving vector  $V$  of the macroblocks. Reverse algorithm is used for decoding.

特開平11-272847

(43) 公開日 平成11年(1999)10月8日

(51) Int. Cl.<sup>5</sup> 識別記号

G 0 6 T 1/00

H 0 4 N 11/04

F I

G 0 6 F 15/66

H 0 4 N 11/04

G 0 6 F 15/66

3 3 0 P

Z

3 1 0

審査請求 未請求 請求項の数17 O L (全 41 頁)

(21) 出願番号 特願平10-74719

(22) 出願日 平成10年(1998) 3月23日

(71) 出願人 000002369

セイコーエプソン株式会社

東京都新宿区西新宿2丁目4番1号

(72) 発明者 保科 彰治

長野県諏訪市大和3丁目3番5号 セイコーエプソン株式会社内

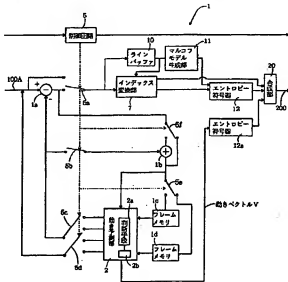
(74) 代理人 弁理士 鈴木 喜三郎 (外2名)

(54) 【発明の名称】 マルチカラー画像からなる動画像の符号化装置およびその方法並びにマルチカラー画像からなる動画像の復号化装置およびその方法

## (57) 【要約】

【課題】 マルチカラー画像からなる動画像の符号化や復号化を行う際にも動き補償を適用できるようにする。また、動き補償の際のブロックの同一性の判断を厳しくしたいが、動きベクトルの数を増やしたくないという相反する目的を同時に達成する。

【解決手段】 このマルチカラー画像からなる動画像の符号化装置1は、動画像を形成する際、動き補償を行う。そして、判断手段2aによって両ブロックの対応する位置の画素が異なる場合の数を算出し、その不一致数が所定数以下のときには同一または類似のブロックと判断し、動きベクトルVを作成している。また、判断手段2aは、各分割されたブロック毎に動きベクトルVを用いるか否かの判断を行い、両ブロックの不一致数が所定数以下のときに同一または類似のブロックと判断し、マクロブロックの動きベクトルVを利用して動き補償を行うようにしている。なお、復号化の際は逆となるアルゴリズムを使用している。



## 【特許請求の範囲】

【請求項1】 インデックスコードが付与されたカラー画像データからなる静止画のフレームを連続させて動画画像を形成する際、符号化対象の着目フレームをそれぞれ $N \times N$ ピクセル ( $N$ は4以上の整数) からなる複数のブロックに分割し、その着目フレーム中のブロックが参照フレーム中のブロックと同一または類似しているとき、着目フレーム中のブロック位置と参照フレーム中のブロック位置との相対変位を動きベクトルとして動き補償を行うマルチカラー画像からなる動画画像の符号化装置において、上記動きベクトルを用いるか否かの判断手段を設け、この判断手段によって両ブロックの対応する位置の画素が異なる場合の数を算出し、その不一致数が所定数以下のときには同一または類似のブロックと判断し、動きベクトルを作成することを特徴とするマルチカラー画像からなる動画画像の符号化装置。

【請求項2】 前記ブロックは、 $MN \times MN$ ピクセル ( $M$ は2以上の整数) からなるより大きなマクロブロックを小ブロック化手段によって $M \times M$ に分割され、前記判断手段は、各分割されたブロック毎に前記動きベクトルを用いるか否かの判断を行うために、前記参照フレーム内の対応するブロックの対応する位置にある画素と当該ブロックの画素とが異なる場合の数を算出し、その不一致数が所定数以下のときに同一または類似のブロックと判断し、上記マクロブロックの動きベクトルを利用して前記動き補償を行うようにしたことを特徴とする請求項1記載のマルチカラー画像からなる動画画像の符号化装置。

【請求項3】 前記ブロックを $8 \times 8$ ピクセルからなるブロックとし、前記所定数を0～5の整数値のいずれか1つとしたことを特徴とする請求項1または2記載のマルチカラー画像からなる動画画像の符号化装置。

【請求項4】 前記動きベクトルを用いなく判断したブロックを、前記着目フレームの符号化時の走査方向と直角となる方向に拾っていくと共に、着目画素の周りの参照画素を利用して当該各ブロック内の各画素を符号化することを特徴とする請求項1、2または3記載のマルチカラー画像からなる動画画像の符号化装置。

【請求項5】 インデックスコードが付与されたカラー画像データからなる静止画のフレームを連続させて動画画像を形成する際、符号化対象の着目フレームをそれぞれ $N \times N$ ピクセル ( $N$ は4以上の整数) からなる複数のブロックに分割し、その着目フレーム中のブロックが参照フレーム中のブロックと同一または類似しているとき、着目フレーム中のブロック位置と参照フレーム中のブロック位置との相対変位を動きベクトルとして動き補償を行うマルチカラー画像からなる動画画像の符号化装置において、上記動きベクトルを用いるか否かの判断工程を設け、この判断工程によって両ブロックの対応する位置の画素が異なる場合の数を算出し、その不一致数が所定数

以下のときには同一または類似のブロックと判断することを特徴とするマルチカラー画像からなる動画画像の符号化方法。

【請求項6】 前記ブロックは、 $MN \times MN$ ピクセル ( $M$ は2以上の整数) からなるより大きなマクロブロックを小ブロック化工程によって $M \times M$ の数の分割され、前記判断工程は、各分割されたブロック毎に前記動きベクトルを用いるか否かの判断を行うために、前記参照フレーム内の対応するブロックの対応する位置にある画素と当該ブロックの画素とが異なる場合の数を算出し、その不一致数が所定数以下のときに同一または類似のブロックと判断し、上記マクロブロックの動きベクトルを利用して前記動き補償を行うようにしたことを特徴とする請求項5記載のマルチカラー画像からなる動画画像の符号化方法。

【請求項7】 前記ブロックは、 $MN \times MN$ ピクセル ( $M$ は2以上の整数) からなるより大きなマクロブロックを小ブロック化手段によって $M \times M$ の数の分割され、前記判断工程は、各分割されたブロック毎に前記動きベクトルを用いるか否かの判断を行うために、前記参照フレーム内の対応するブロックの対応する位置にある画素と当該ブロックの画素とが異なる場合の数を算出し、その不一致数が所定数以下のときに $N \times N$ ピクセルのブロックについて同一または類似のブロックと判断し、上記マクロブロックの動きベクトルを利用して前記動き補償を行うようにしたことを特徴とする請求項5記載のマルチカラー画像からなる動画画像の符号化方法。

【請求項8】 前記ブロックを $8 \times 8$ ピクセルからなるブロックとし、前記所定数を0～5の整数値のいずれか1つとしたことを特徴とする請求項5、6または7記載のマルチカラー画像からなる動画画像の符号化方法。

【請求項9】 前記動きベクトルを用いなく判断したブロックを、前記着目フレームの符号化時の走査方向と直角となる方向に拾っていくと共に、前記着目画素の周りの参照画素を利用して当該各ブロック内の各画素を符号化することを特徴とする請求項5、6、7または8記載のマルチカラー画像からなる動画画像の符号化方法。

【請求項10】 入力される対象符号化データをインデックスコードが付与されたカラー画像データに復号するエンコード-復号化手段と、該カラー画像データからなる静止画を連続させて動画画像を形成する動画画像形成手段とを備え、動画画像形成手段は、復号化対象の着目フレームをそれぞれ $N \times N$ ピクセル ( $N$ は4以上の整数) からなる複数のブロックに分割し、その着目フレーム中のブロックに関する復号化データ中にそのブロックの相対変位の方向を示す動きベクトルが存在するとき、参照フレーム中の同一または類似のブロックを利用してそのブロックを復号する動き補償を行うマルチカラー画像からなる動画画像の復号化装置において、上記動きベクトルは、上記着目フレーム中のブロックと上記参照フレーム中の

対応するブロックの対応する位置の画素が異なる場合のその不一致数が所定数以下のときに発生させられたものであることを特徴とするマルチカラー画像からなる動画像の復号化装置。

【請求項11】 前記ブロックを、 $MN \times MN$ ピクセル ( $M$ は2以上の整数) からなるより大きなマクロブロックから小ブロック化手段によって  $M \times M$  の数に分割し、上記マクロブロックの動きベクトルを利用して上記小さな各ブロックを復号するか否かを判断する判断手段を有していることを特徴とする請求項10記載のマルチカラー画像からなる動画像の復号化装置。

【請求項12】 前記ブロックを、 $8 \times 8$ ピクセルからなるブロックとし、前記所定数を0～5の整数値のいずれか1つとしたことを特徴とする請求項10または11記載のマルチカラー画像からなる動画像の復号化装置。

【請求項13】 前記動きベクトルを有しないブロックを、前記着目フレームの符号化時の走査方向と直角となる方向にたがわせて復号すると共にその復号の際、着目画素の周りの参照画素を利用して当該動きベクトルを有さないブロック内の各画素を復号化することを特徴とする請求項10、11または12記載のマルチカラー画像からなる動画像の復号化装置。

【請求項14】 入力される対象符号化データをインデックスコードが付与されたカラー画素データに復号するエントロピー復号化工程と、該カラー画素データからなる静止画を連続させて動画像を形成する動画像形成工程とを備え、動画像形成工程は、復号化対象の着目フレームをそれぞれ  $N \times N$ ピクセル ( $N$ は4以上の整数) からなる複数のブロックに分割し、その着目フレーム中のブロックに関する復号化データ中にそのブロックの相対変位の方向を示す動きベクトルが存在すると、参照フレーム中の同一または類似のブロックを利用してそのブロックを復号する動き補償を行うマルチカラー画像からなる動画像の復号化方法において、上記動きベクトルは、上記着目フレーム中のブロックと上記参照フレーム中の対応するブロックの対応する位置の画素が異なる場合のその不一致数が所定数以下のときに発生させられたものであることを特徴とするマルチカラー画像からなる動画像の復号化方法。

【請求項15】 前記ブロックを、 $MN \times MN$ ピクセル ( $M$ は2以上の整数) からなるより大きなマクロブロックから小ブロック化工程によって  $M \times M$  の数に分割し、上記マクロブロックの動きベクトルを利用して上記小さな各ブロック復号するか否かを判断する判断工程を有していることを特徴とする請求項14記載のマルチカラー画像からなる動画像の復号化方法。

【請求項16】 前記ブロックを、 $8 \times 8$ ピクセルからなるブロックとし、前記所定数を0～5の整数値のいずれか1つとしたことを特徴とする請求項14または15記載のマルチカラー画像からなる動画像の復号化方法。

【請求項17】 前記動きベクトルを有しないブロックを、前記着目フレームの符号化時の走査方向と直角となる方向にたがわせて復号すると共にその復号の際、着目画素の周りの参照画素を利用して当該動きベクトルを有さないブロック内の各画素を復号化することを特徴とする請求項14、15または16記載のマルチカラー画像からなる動画像の復号化方法。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、マルチカラー画像からなる動画像の符号化装置およびその方法ならびにマルチカラー画像からなる動画像の復号化装置およびその方法に関する。そして、さらに詳細に述べれば、マルチカラー画像を符号化および復号化する際の動き補償の改良に関する。

【0002】

【従来の技術】従来からパソコンやゲーム機器等では、マルチカラー画像と呼ばれる画像が使用されている。このマルチカラー画像とは、代表色画像とか限定色画像等とも呼ばれているもので、図4に示すように、特定の色、すなわち特定のR(赤)、G(緑)、B(青)の値を持つ色に対してインデックスを付与し、そのインデックスのデータを利用して、16色や256色等の限定された代表色で表現するようにした画像のことである。

【0003】このようなマルチカラー画像のデータは、仮にR、G、Bの各色が8ビット(256種)で表されるとしたら、合計24ビット必要になるのであるが、インデックスそのものも例えば8ビットで表示するようにしているので、相当な圧縮率となっている。しかし、圧縮はされているが、それでも情報量が多いため、何の工夫もせず、そのままの形で処理すると、メモリ容量が大きくなり、また通信速度も遅くなり実用的でない。したがって、マルチカラー画像も他の画像データと同様にその圧縮技術は極めて重要なものとなる。特に、マルチカラー画像は、その色の数が限定されていることから、ロスレスでの符号化および復号化、すなわち可逆的な圧縮技術が必要とされている。

【0004】一方、近年、データ圧縮の手法の一つとして、エントロピー符号器および復号器を用いた技術が注目されている。このエントロピー符号化および復号化技術の一つとして、例えば、算術符号化および復号化の技術を用いたものがある。特開昭62-185413号公報、特開昭63-74324号公報、特開昭63-76525号公報等に記載されている。

【0005】図35に、このような技術を用いた従来のマルチカラー画像の符号化システム50および復号化システム60を示す。この符号化システム50は、ラインバッファ51と、エントロピー符号器52を含むものである。入力されるインデックスのデータ、すなわちカラー画素データ100Aは、ラインバッファ51および

エントロピー符号器52へ入力される。このカラー画像データ100Aは、図36に示すように、いずれもラスターキャンセル毎に水平走査順に順次画像データとして入力される。なお、このインデックスのデータ、すなわちカラー画像データ100Aを作成する方法としては、入力する色の順番にインデックスを付与する方法が一般的であり、図41に示すように、インデックスの番号が近いもの、例えば「1」と「2」でもその色が大きく異なったり、インデックスの番号が遠いもの、例えば「100」と「200」でもその色は近似している現象が生じている。このような現象を避けるため、特開平5-328142に示されるように、色の近いものに連続番号を付与するようにしたものも現れている。

【0006】符号化システム50中のラインバッファ51は、参照画像生成手段として、既に入力されたカラー画像データ100Aから、符号化対象画像Xに対する参照画像データA、B、C、Dを作成する。すなわち、ラインバッファ51は、画像をスキャンするときにnライン(1~5ライン程度が多い)分の履歴を記憶しておく。そして、符号化対象画像Xのカラー画像データ100Aが入力されるごとに、この直前の画像Aと、周辺の画像B、C、Dとからなる一連の画像データを参照画像データ110としてエントロピー符号器52へ向けて出力する。

【0007】このエントロピー符号器52は、例えば、算術符号化またはハフマン符号化などの手法を用いて形成される。そして、参照画像データ110を状態信号として用い、対象カラー画像データ100Aを符号化データ200に変換出力する。

【0008】一方、復号化システム60は、ラインバッファ61とエントロピー復号器62を含んで構成される。ここにおいて、ラインバッファ61とエントロピー復号器62は、入力される符号化データ200を符号化システム50のラインバッファ51、エントロピー符号器52とは全く逆の手順で復号化出力するように形成されている。

【0009】このようにして、符号化システム50と、復号化システム60とは、互いに全く逆のアルゴリズムを用いて、カラー画像データ100Aを符号化データ200に符号化し、さらにこの符号化データ200をカラー画像データ100Bに復号化して出力することができる。したがって、このシステムは、各種用途に幅広く用いることができる。

【0010】ところで、このようなシステムでは、カラー画像データ100Aの値、すなわちインデックスの番号が一定の番号付近に偏るとそのデータの圧縮率が向上する。また、このシステムでは、参照画像データ110を、エントロピー符号器52、エントロピー復号器62の状態信号として使用している。したがって、その状態数、すなわち、参照画像数を多くとれば、同様にデータ

圧縮率は向上する。すなわち、算術符号化またはハフマン符号化などの手法を用いてエントロピー符号器52および復号器62を構成する場合には、0または1のシンボルの発生確率に大きな偏りがあると、データの圧縮率を高めることができる。これは、エントロピー符号化技術では、発生確率の高い入力データには短い符号化データを割り当て、発生率の低い入力データには相対的に長い符号化データを割り当てるからである。

【0011】シンボル、すなわちインデックスの番号の発生確率の大きな偏りを得るために、従来より、入力データをいくつかの状態に分類して符号化することが行われている。なぜなら、分類をしないと、良い圧縮率は得られないからである。例えば、図35に示す従来の手法では、ラインバッファ51、61を用い、参照画像データを作成し、これを分類用の状態信号としてエントロピー符号器52およびエントロピー復号器62へ入力している。そして、これらエントロピー符号器52およびエントロピー復号器62は、前記状態信号を用いることにより入力データを分類し、符号化および復号化を行っている。すなわち、参照画像データの各状態の発生確率を計算し、その発生確率の高い組み合わせのものに短い符号化データを割り当てている。そして、これによりデータの圧縮率を高めている。

【0012】しかし、前述したエントロピー符号器52およびエントロピー復号器62では、参照画像データの状態数に対応した数の符号化パラメータテーブルが必要となる。このため、圧縮率を高めるために参照画像数を大きくとればよいと、符号化および復号化のパラメータテーブルが大きくなる。このため、エントロピー符号器52およびエントロピー復号器62が大型化かつ高価となってしまうという問題がある。

【0013】例えば、カラー画像データ、すなわちインデックスの番号を4ビットデータ(16種)で構成し、しかも参照画像データ110の画像数が4である場合を想定する。この場合には、符号化および復号化パラメータテーブルの状態数は、4画像×4ビット=16ビット分の状態数、すなわち $2^{16}$ の状態数をとる。このため、 $2^{16}=65536$ 通りのパラメータテーブルを用意しなければならない。このことから、参照画像を1つ増やすごとに、その分、符号化および復号化パラメータテーブルが極めて大きくなり、エントロピー符号器52およびエントロピー復号器62を構成するハードウェアが大型化してしまうことが理解される。しかも、対象画像も4ビット、すなわち、4プレーンで構成され、各プレーンに1ビットずつの信号が割り当てられ、結果として4ビットで16通りの値(色)をとるため、パラメータテーブルは $65536 \times 16$ の大きさを持つテーブルとなる(図37参照)。

【0014】このような問題に対し、対象画像のカラーシンボル、すなわち色に対応するインデックスの番号の

出現頻度の偏りを計算し、出現頻度順位に対応して、インデックスの番号を並び替える色順位変換の方法（特開平6-276041号）がある。すなわち、これにより出現頻度の高いものに短い符号化データを割り当て、さらに、圧縮率を高めている。また、この公開報には、エントロピー符号器52およびエントロピー復号器62の中に縮退した状態数に応じてパラメータテーブルを小さくさせる技術も開示されている。

【0015】この特開平6-276041号公報に示されている状態数を縮退するシステムの特徴は、図38に示すように、従来の符号化システム50や復号化システム60と同様にエントロピー符号器52およびエントロピー復号器62に参照画素データ110を状態信号として入力するわけであるが、その入力に際し、その状態信号140を、ラインバッファ51、61から出力される参照画素データ110を縮退する状態縮退器53、63によって生成する点にある。

【0016】この状態縮退器53、63は、入力される参照画素データ110を、より少ないビット数の状態信号140に縮退し、対応するエントロピー符号器52およびエントロピー復号器62へ向け出力するように構成されている。なお、予測器54、64は、それぞれカラーシンボルの出現頻度に基づいてカラー画素データを色順位に変換するためおよびその逆を行うための色順位テーブル（詳細は後述）をそのメモリに保有しているものである。

【0017】なお、縮退とは、縮退後の状態数に、元の状態を分類する操作である。この分類は、分類後のエントロピー（1つのシンボルを表示するための平均情報量）が最少となるように、その組み合わせを選択して行う。そして、縮退後の状態数、すなわち、分類された後の状態数に対して識別ビットを付加する。これが状態信号140である。

【0018】ところで、状態縮退器53、63に用いる縮退テーブルとしては、参照画素データ110のカラーシンボルの組み合わせパターンと、縮退データとの関係を特定する縮退テーブルを設定し、この縮退テーブルを用い、入力される参照画素データ110のカラーシンボルの組み合わせパターンを、縮退データに変換出力する方法がある。

【0019】図39には、このような手法を用いて行われる縮退動作の一例が示されている。ここでは、説明を簡単にするために、図39（A）に示すよう、符号化対象画素Xに対し、A、B、Cの3つの画素から形成されるマルコフモデルを参照画素パターンとして用いる場合を例にとり説明する。

【0020】参照画素が、図39（A）に示すように、3つの画素から構成される場合には、そのカラーシンボルの組み合わせパターンは、図39（B）に示すように5通りとなる。すなわち、3つに画素のカラーシンボル

が全て一致するパターンと、2つのカラーシンボルのみが一一致する場合に該当する3つのパターンと、全ての画素のカラーシンボルが異なるパターンの計5つのパターンに分類される。

【0021】したがって、図39（B）に示すテーブルを状態縮退器53、63の縮退テーブルとして用いることにより、本来3つの画素の組み合わせが取りうる2<sup>3</sup>

パターンの状態を、図39（B）に示す5つの状態S1～S5に縮退することができる。このようにすることによって、参照画素データ110を効果的に縮退し、エントロピー符号器52およびエントロピー復号器62の状態数を大幅に少なくすることができる。

【0022】ところで、このような算術符号化および復号化の一般的な手法は、既に1画像符号化標準JBIG（インターナショナルスタンダードISO/IEC1544）のp26～44およびp44～p50に詳細に述べられているが、ここでは後述する本発明を展開する際の前提技術として簡単に説明する。

【0023】図35に用いられる算術符号型のエントロピー符号器52の一例を図40に示す。なお、算術復号型のエントロピー復号器62の構成は、エントロピー符号器52の構成と実質的に同一であるので、ここではその説明は省略する。

【0024】このエントロピー符号器52は、算術演算部55と、状態記憶器として機能する発生確率生成手段56とを含んで構成される。この発生確率生成手段56内には、符号化に必要なシンボル発生確率を決定するために必要な状態パラメータテーブルが書き込まれている。上記の状態パラメータは、入力される状態信号によって特定される。そして、この状態信号によって特定された状態パラメータのテーブルに対し、発生確率生成手段56の発生確率演算パラメータが算術演算部55へ向けて出力される。算術演算部55は、このようにして入力される発生確率に基づき、エントロピー符号化を行い、入力される色順位データ120を符号化データ200に変換出力する。そして、符号化した色順位データ120の値により、状態信号に対する発生確率を再計算し、演算パラメータ更新値として、発生確率生成手段56へ入力する。この更新結果が次データの発生確率としてテーブルに記憶されることで、エントロピー符号器52の圧縮効率が向上することとなる。

【0025】なお、色順位データ120を生成するために、先に述べたように予測器54、64内に色順位テーブルが配置されている。この色順位テーブルの一例として、図42に示されるものが知られている（特開平6-276041号公報参照）。この例では、符号化対象画素Xに対しての色順位テーブルを決める際、2次元的な周辺画素データR0、R1、R2、R3を上位の色順位データとして使用し、符号化対象画素Xと同一ラインの1次元テーブルを下位の色順位データとして使用するも

のである。このとき、1次元テーブルから周辺画素データR0, R1, R2, R3のカラーシンボルを除去した後、上位と下位の色順位データをドッキングさせ符号化対象画素Xの色順位テーブルとしている。

【0026】具体的にどのように色順位テーブルができるかを図42に基づき説明する。符号化されるカラーシンボルが16色の場合を考える。仮に、色順位を図42(A)に示すように、各画素の位置R0, R1, R2...R8...で固定したとき、それぞれのカラーシンボルが図42(B)に示すように、C4, C3, C6, C5, C2, C2...のとき、できあがる最新出現表となる色順位テーブルは、図42(C)に示すようになる。すなわち、最上位はR0のC4, 2番目はR1のC3, 3番目はR2のC6, 4番目はR3のC5, 5番目はR4のC2, 6番目はR5にあるC2となるが、C2は既に発生しており、さらにR6のC4も既に発生しているの、第6番目はR7のC0となる。このようにして既に上位にある色すなわち、R0~R3に出現するカラーシンボルを除いた色順位データがR0~R3のデータに加わり、16色のカラーシンボルの第1番目から16番目までが決められる。なおこのとき、上位4つの周辺画素を学習により可変とすることもできる。

【0027】一方、動画像の符号化や復号化において、動き補償と呼ばれるものが知られている。この動き補償は、動画が静止画の連続であり、その1枚1枚を見ると、前のフレームや後のフレームと相関があることに着目したものである。具体的に、着目フレームを16×16ピクセルの複数のブロックに分割し、当該着目フレーム中の所定ブロックが参照フレームの1ピクセル左にずれた同様のブロックと似ている場合、着目フレーム中のそのブロックの画像データを符号化または復号化せず、参照フレーム中の1ピクセル左にずれたブロックに似ているというデータを符号化または復号化するものである。

【0028】すなわち、この動き補償では、着目フレームの全データを符号化したり復号化するのではなく、変化のあった部分だけのデータを送ると共に、参照フレーム中のブロックと同一となるブロックについては、さらに動きベクトルをデータ化することによってデータ量のさらなる減少が可能となっている。

【0029】動き補償における動きベクトルを算出する場合、R、G、Bからなる自然画のときには、着目フレームと参照フレームの対応するブロックのR、G、Bの差分である $\Delta R$ ,  $\Delta G$ ,  $\Delta B$ を算出する。そして、その平均2乗誤差 $(=\Delta R^2 + \Delta G^2 + \Delta B^2)$ を得て、この値が一定値以下のときに同一ブロックと判断し、動きベクトルを生成している。また、その他に、着目フレームと参照フレームの対応するブロックの対応する位置の画素の差分の絶対値和を計算し、その最小値を求め、動きベクトルを得る方法も知られている。

#### 【0030】

【発明が解決しようとする課題】マルチカラー画像の場合、図41に示されるようにインデックスの番号が近いものでもその色が大きく異なったり、インデックスの番号が遠いものでもその色が近似しているという状態が生ずる。このため、マルチカラー画像に動き補償を適用しようとしても、従来の平均2乗誤差方式や絶対値和方式では、適切にその同一性を判断できない。

【0031】特に、色順位テーブルを利用して符号化、復号化を行うものの場合、インデックスの番号が常に変えられ続けるため、動き補償を行わせる際、上述の平均2乗誤差方式等の手法は全く採用できないこととなる。

【0032】さらに、マルチカラー画像の場合、隣接するカラーシンボルが大きく異なる場合が多く、かつその変化が目立ちやすい。このため、動き補償を行うに当たり、ブロックの同一性の判断を厳しくする必要がある。ところが、この判断を厳しくすると大きなブロック間では、同一と判断されるものが極端に少なくなり、動き補償によるデータ圧縮の効果が生じなくなってしまう。一方、小さなブロックに分割して、動きベクトルを各ブロックで得るようになると、動きベクトルのデータ量が増え、効果的な圧縮ができなくなってしまう。

【0033】本発明は、以上のような問題に対処してなされたものであり、マルチカラー画像からなる動画像の符号化や復号化を行う際にも動き補償を適用できるようにしたマルチカラー画像からなる動画像の符号化装置およびその方法ならびにマルチカラー画像からなる動画像の復号化装置およびその方法を提供することを目的とする。また、本発明は、動き補償の際のブロックの同一性の判断を厳しくしたいが、動きベクトルの数を増やしたくないという相反する目的を同時に達成できるマルチカラー画像からなる動画像の符号化装置等を提供することを目的とする。

#### 【0034】

【課題を解決するための手段】かかる目的を達成するため、請求項1記載の発明では、インデックスコードが付与されたカラー画素データからなる静止画のフレームを連続させて動画像を形成する際、符号化対象の着目フレームをそれぞれN×Nピクセル(Nは4以上の整数)からなる複数のブロックに分割し、その着目フレーム中のブロックが参照フレーム中のブロックと同一または類似しているとき、着目フレーム中のブロック位置と参照フレーム中のブロック位置との相対位置を動きベクトルとして動き補償を行うマルチカラー画像からなる動画像の符号化装置において、動きベクトルを用いるか否かの判断手段を設け、この判断手段によって両ブロックの対応する位置の画素が異なる場合の数を算出し、その不一致数が所定数以下のときには同一または類似のブロックと判断し、動きベクトルを作成している。

【0035】このため、マルチカラー画像からなる動



像においても、自然画と同様に動き補償が使用できることとなり、データ圧縮が可能となる。

【0036】また、請求項2記載の発明では、請求項1記載のマルチカラー画像からなる動画像の符号化装置において、ブロックは、 $M \times M$ ピクセル ( $M$ は2以上の整数) からなるより大きなマクロブロックを小ブロック化手段によって  $M \times M$  に分割され、判断手段は、各分割されたブロック毎に前記動きベクトルを用いるか否かの判断を行うために、参照フレーム内の対応するブロックの対応する位置にある画素と当該ブロックの画素とが異なる場合の数を算出し、その不一致数が所定数以下のときに同一または類似のブロックと判断し、マクロブロックの動きベクトルを利用して動き補償を行うようにしている。

【0037】このように、より大きなブロック間で動きベクトルを作成するか否かを判断し、さらに小さいブロック間でも動き補償を行うか否かを判断しているため、動きベクトルの作成数を増やすことなく動き補償の要否を厳格に判断することが可能となる。すなわち、マルチカラー画像の場合でも、データ圧縮を効果的に行えるようになる。

【0038】さらに、請求項3記載の発明では、請求項1または2記載のマルチカラー画像からなる動画像の符号化装置において、ブロックを  $8 \times 8$  ピクセルからなるブロックとし、所定数を0〜5の整数値のいずれか1つとしている。このため、一般的な動き補償の際の  $16 \times 16$  ピクセルのマクロブロックに比べ、効果的な動き検出が可能となる。しかも、判定基準が厳しいので、マルチカラー画像であっても画像の乱れは生じにくくなる。

【0039】加えて、請求項4記載の発明では、請求項1、2または3記載のマルチカラー画像からなる動画像の符号化装置において、動きベクトルを用いないと判断したブロックを、着目フレームの符号化時の走査方向と直角となる方向に拾っていくと共に、着目画素の周りの参照画素を利用して当該各ブロック内の各画素を符号化している。このように、動きベクトルを用いないブロックを、着目フレームの符号化時の走査方向と直角となる方向に拾っていく符号化するので、メモリが少なくて済むと共に、メモリへのアクセスが少なくなり符号化速度が向上する。

【0040】また、請求項5記載の発明では、インデックスコードが付与されたカラー画素データからなる静止面のフレームを連続させて動画像を形成する際、符号化対象の着目フレームをそれぞれ  $N \times N$  ピクセル ( $N$ は4以上の整数) からなる複数のブロックに分割し、その着目フレーム中のブロックが参照する参照フレーム中のブロックと同一または類似しているとき、着目フレーム中のブロック位置と参照フレーム中のブロック位置との相対変位を動きベクトルとして動き補償を行うマルチカラー画像からなる動画像の符号化方法において、動きベ

クトルを用いるか否かの判断を行う判断工程を設け、この判断工程によって両ブロックの対応する位置の画素が異なる場合の数を算出し、その不一致数が所定数以下のときには同一または類似のブロックと判断している。このため、マルチカラー画像からなる動画像においても、自然画と同様に動き補償が使用できることとなり、データ圧縮が可能となる。

【0041】さらに、請求項6記載の発明は、請求項5記載のマルチカラー画像からなる動画像の符号化方法において、ブロックは、 $M \times M$ ピクセル ( $M$ は2以上の整数) からなるより大きなマクロブロックを小ブロック化工程によって  $M \times M$  の数に分割され、判断工程は、各分割されたブロック毎に動きベクトルを用いるか否かの判断を行うために、参照フレーム内の対応するブロックの対応する位置にある画素と当該ブロックの画素とが異なる場合の数を算出し、その不一致数が所定数以下のときに同一または類似のブロックと判断し、マクロブロックの動きベクトルを利用して動き補償を行うようにしている。

【0042】また、請求項7記載の発明は、請求項5記載のマルチカラー画像からなる動画像の符号化方法において、ブロックは、 $M \times M$ ピクセル ( $M$ は2以上の整数) からなるより大きなマクロブロックを小ブロック化工程によって  $M \times M$  の数に分割され、判断工程は、各分割されたブロック毎に動きベクトルを用いるか否かの判断を行うために、参照フレーム内の対応するブロックの対応する位置にある画素と当該ブロックの画素とが異なる場合の数を算出し、その不一致数が所定数以下のときに  $N \times N$  ピクセルのブロックについて同一または類似のブロックと判断し、マクロブロックの動きベクトルを利用して前記動き補償を行うようにしている。

【0043】このように、請求項6または7記載の発明では、より大きなブロック間で動きベクトルを作成するか否かを判断し、さらに小さいブロック間でも動き補償を行うか否かを判断しているため、動きベクトルの作成数を増やすことなく動き補償の要否を厳格に判断することが可能となる。すなわち、マルチカラー画像の場合でも、データ圧縮を効果的に行えるようになる。

【0044】また、請求項8記載の発明は、請求項5、6または7記載のマルチカラー画像からなる動画像の符号化方法において、ブロックを  $8 \times 8$  ピクセルからなるブロックとし、所定数を0〜5の整数値のいずれか1つとしている。このため、一般的な動き補償の際の  $16 \times 16$  ピクセルのマクロブロックに比べ、効果的な動き検出が可能となる。しかも、判定基準が厳しいので、マルチカラー画像であっても画像の乱れは生じにくくなる。

【0045】加えて、請求項9記載の発明は、請求項5、6、7または8記載のマルチカラー画像からなる動画像の符号化方法において、動きベクトルを用いないと判断したブロックを、着目フレームの符号化時の走査方

向と直角となる方向に拾っていくと共に、着目画素の周りの参照画素を利用して当該各ブロック内の各画素を符号化している。このように、動きベクトルを用いないブロックを、着目フレームの符号化時の走査方向と直角となる方向に拾っていき符号化するので、メモリが少なくて済むと共に、メモリへのアクセスが少なくなり符号化速度が向上する。

【0046】また、請求項10記載の発明では、入力される対象符号化データをインデックスコードが付与されたカラー画素データに復号するエントロピー復号化手段と、該カラー画素データからなる静止画を連続させて動画像を形成する動画像形成手段とを備え、動画像形成手段は、復号化対象の着目フレームをそれぞれ $N \times N$ ピクセル ( $N$ は4以上の整数) からなる複数のブロックに分割し、その着目フレーム中のブロックに関する復号化データ中にそのブロックの相対変位の方法を示す動きベクトルが存在するとき、参照フレーム中の同一または類似のブロックを利用してそのブロックを復号する動き補償を行うマルチカラー画像からなる動画像の復号化装置において、動きベクトルは、着目フレーム中のブロックと参照フレーム中の対応するブロックの対応する位置の画素が異なる場合のその不一致数が所定数以下のときに発生させられたものとしている。

【0047】このため、マルチカラー画像からなる動画像の復号化においても、自然画と同様に動き補償が使用でき、復号化装置のメモリ容量を小さくできると共にデータ復号の速度を向上させることができる。

【0048】さらに、請求項11記載の発明は、請求項10記載のマルチカラー画像からなる動画像の復号化装置において、ブロックを、 $MN \times MN$ ピクセル ( $M$ は2以上の整数) からなるより大きなマクロブロックから小ブロック化手段によって $M \times M$ の数に分割し、マクロブロックの動きベクトルを利用して小さな各ブロックを復号するか否か判断する判断手段を有している。このため、より大きなマクロブロック間の動きベクトルを利用して小さなブロックの動き補償を行うことが可能になるので、動きベクトルの数を増やすことなく、小さなブロックについても動き補償が可能となる。このため、マルチカラー画像の復号化の場合でも、データのメモリ容量を小さくできると共にデータ復号を効率的に行えるものとなる。

【0049】また、請求項12記載の発明では、請求項10または11記載のマルチカラー画像からなる動画像の復号化装置において、ブロックを、 $8 \times 8$ ピクセルからなるブロックとし、所定数を0～5の整数値のいずれか1つとしている。このため、一般的な動き補償の際の $16 \times 16$ ピクセルのマクロブロックに比べ、効果的な動き復号化が可能となる。しかも、判定基準が厳しいので、マルチカラー画像であっても画像の乱れは生じにくくなる。

【0050】加えて、請求項13記載の発明では、請求項10、11または12記載のマルチカラー画像からなる動画像の復号化装置において、動きベクトルを有しないブロックを、着目フレームの符号化時の走査方向と直角となる方向につなぎ合わせていき復号するので、メモリが少なくて済むと共に、メモリへのアクセスが少なくなり復号化速度が向上する。

【0051】また、請求項14記載の発明では、入力される対象符号化データをインデックスコードが付与されたカラー画素データに復号するエントロピー復号化工程と、該カラー画素データからなる静止画を連続させて動画像を形成する動画像形成工程とを備え、動画像形成工程は、復号化対象の着目フレームをそれぞれ $N \times N$ ピクセル ( $N$ は4以上の整数) からなる複数のブロックに分割し、その着目フレーム中のブロックに関する復号化データ中にそのブロックの相対変位の方法を示す動きベクトルが存在するとき、参照する参照フレーム中の同一または類似のブロックを利用してそのブロックを復号する動き補償を行うマルチカラー画像からなる動画像の復号化方法において、動きベクトルは、着目フレーム中のブロックと参照フレーム中の対応するブロックの対応する位置の画素が異なる場合のその不一致数が所定数以下のときに発生させられたものとしている。

【0052】このため、マルチカラー画像からなる動画像の復号化においても、自然画と同様に動き補償が使用でき、この方法を採用すると、復号化装置のメモリ容量を小さくできると共にデータ復号の速度を向上させることができる。

【0053】さらに、請求項15記載の発明は、請求項13記載のマルチカラー画像からなる動画像の復号化方法において、ブロックを、 $MN \times MN$ ピクセル ( $M$ は2以上の整数) からなるより大きなマクロブロックから小ブロック化工程によって $M \times M$ の数に分割し、マクロブロックの動きベクトルを利用して小さな各ブロックを復号するか否か判断する判断手段を有している。このため、マルチカラー画像の復号化の場合でも、データのメモリ容量を小さくできると共にデータ復号を効率的に行えるものとなる。

【0054】加えて、請求項16記載の発明は、請求項14または15記載のマルチカラー画像からなる動画像の復号化方法において、ブロックを、 $8 \times 8$ ピクセルからなるブロックとし、所定数を0～5の整数値のいずれか1つとしている。このため、一般的な動き補償の際の $16 \times 16$ ピクセルのマクロブロックに比べ、効果的な動き復号化が可能となる。しかも、判定基準が厳しいの

で、マルチカラー画像であっても画像の乱れは生じにくくなる。

【0055】また、請求項17記載の発明は、請求項14、15または16記載のマルチカラー画像からなる動画像の復号化方法において、動きベクトルを有しないブロックを、着目フレームの符号化時の走査方向と直角となる方向につなぎ合わせて復号すると共にその復号の際、着目画素の周囲の参照画素を利用して当該動きベクトルを有さないブロック内の各画素を復号化している。このように、動き補償を行わないブロックを、着目フレームの符号化時の走査方向と直角となる方向につなぎ合わせて復号するので、メモリが少なくて済むと共に、メモリへのアクセスが少なくなり復号速度が向上する。

【0056】

【発明の実施の形態】本発明の実施の形態を図1から図34に基づき説明する。なお、従来技術のデータと対応する各データには、同一符号を付して説明する。

【0057】図1に、本発明に係るマルチカラー画像からなる動画像の符号化システム1の好適な実施の形態を示す。また、図3に、図1の符号化システム1に対応するマルチカラー画像からなる動画像の復号化システム3の好適な実施の形態を示す。

【0058】この符号化システム1は、動き補償を行うための動きベクトルVを生成する動き予測部2と、入力されるカラー画素データ100Aから動き予測部2で生成されるデータを減算する減算器1aと、後述するインデックス変換部7に入力するデータと動き予測部2からのデータとを加算する加算器1bと、静止画としてのフレームをメモリする2つのフレームメモリ1c、1dと、各スイッチ5a、5b、5c、5d、5e、5fを制御する制御回路8と、インデックス変換部7と、周辺画素生成手段および参照画素生成手段となるラインバッファ10と、縮退手段となるマルコフモデル生成部11と、エントロピー符号化手段となる2つのエントロピー符号器12、12aと、合成部20とを含み、入力されるカラー画素データ100Aのデータストリームを符号化データ200のデータストリームに変換して出力するように構成されている。

【0059】減算器1aおよび加算器1bによって、フレーム間の予測符号化手段を構成する。これは着目フレームに対して隣接フレームの同一位置の画素データの減算することにより、エントロピー符号器12に入力するデータの値を0近傍の値に取め符号化効率を上昇させる手段である。フレームがRGB(Red, Green, Blue)それぞれの色で表現されている場合には、減算器1aおよび加算器1bによって演算する値は、それぞれ色ごとに演算する。この場合は画素データ値の近傍値が類似した色となっているため、差分値は0の値に集中する。一方、インデックスバレットによって構成さ

れたマルチカラー画像では、画素データ値の近傍が必ずしも類似色ではない。よってフレーム間の予測符号化は、着目フレームと隣接フレームの値が同値、すなわち同色の場合のみ0になる。このため、本発明において、前述のフレーム間予測符号化手段は必須ではない。

【0060】次に、この動き補償の一般的な動作について説明する。カラー画素データ100Aは、マルチカラー画素データとして入力する。減算器1aでは、カラー画素データ100Aから入力する現在の画素データと、フレームメモリ1c、1dから転送する過去または未来の画素データとの差分を行う。これは、フレーム間予測符号化のフレーム間差分の減算器に当たる。なお、マルチカラー画像では差分によってかえって、情報量が増大する場合がある。この場合には、減算器を使用しない。減算を行わない方法としては、図1に示すように、カラー画素データ100Aを減算器1aを迂回する経路に接続する方法の他、減算器を設けない方法、またはフレームメモリ1c、1dからのデータに0情報を送る方法などがある。

【0061】さらに、スイッチ5aを通過したこの情報を、加算器1bに入力するデータと、エントロピー符号器12に入力するデータとに分岐させる。エントロピー符号器12に入力するデータについては後述する。加算器1bは、フレーム間予測符号化のフレーム間差分の加算器に当たる。加算器1bでは、減算器1aで減算した過去または未来のデータと同じデータで加算を行う。よって、この加算器1bを通過した信号は、減算前のカラー画素データ100Aと同一となる。減算器1aと加算器1bは、対にして、フレーム間予測符号化のフレーム間差分を行うので、減算器1aを使用しない場合には、加算器1bは必要なく、スイッチ5fによって使用しない側に切り換えられる。加算を行わないためには、図1に示すように、加算器1bを迂回する経路に信号を切り替える方法の他、フレームメモリ1c、1dからのデータ線に0を入力する方法が採用される。

【0062】この元の信号と同一の信号を、動き予測部2に入力する。元の信号が、ラスタスキャンにしたがって入力されている場合には、動き予測部2にてブロック単位の画素比較を行うために、画像バッファ(図示省略)に画素データを入力する。この場合には、画像バッファは少なくとも、ブロックの縦の画素数×画像フレームの横画素数分の画像データを入力できる容量になる。この画像バッファの値と、フレームメモリ1c、1dに格納された過去または未来の画像データと排他的論理和XORにて、該当画素が互いに一致しているかどうかを検証する。この検証は、画像バッファ中のブロックと同じ座標の、フレームメモリ1c、1d中のブロックを取り出しXOR比較する他に、その同じ座標の周囲のブロックについても比較する。

【0063】画素データは、マルチカラー画像では、1

画素あたり1バイト程度のデータから成り立つ。排他的論理和XORは、フレーム画像データの画素と、画像バッファの画素が同一であるかどうかの確認であるので、1画素を表す全てのビットをXORするのではなく、どれか1ビットでも異なっているものがあれば、確認は終了する。

【0064】画素の一致、不一致は、動き予測部2内のカウンタ（図示省略）によって、カウントする。ここでは不一致検出ごとにカウンタが増大するとする。このカウンタは、フレームメモリ1c、1dの1ブロック分のカウントを終了することにリセットされる。リセット直前のカウンタ値は、カウンタに内在している、カウント値の配列格納部（図示省略）に、現在の比較しているブロックの位置と、フレームメモリ1c、1dのブロック位置とのずれごとに格納する。

【0065】現在の画像の1ブロック分の比較の終了時には、配列格納部中でカウント値のもっとも小さい値を検索する。その値が設定した閾値よりも大きい場合には、現在のブロックが過去または未来の該当位置のブロックと、同一または類似していないと判別する。判別の結果によって行う動作は、順方向予測すなわち過去から現在方向の予測と、逆方向予測すなわち未来から現在方向の予測と、双方向予測すなわち過去または未来から現在への予測とでは動作が異なる。

【0066】順方向予測では、同一または類似であると判別した時、過去フレームの最も不一致点が少ない領域を、現在のフレームへコピーする。逆に同一または類似ではない時には、画像バッファから現在のフレームにコピーする。このとき、動き予測部2以外の部分では、画像バッファの比較したブロックに相当するデータをエントロピー符号器12によって符号化する。

【0067】逆方向予測では、同一または類似であると判別した時、未来フレームの最も不一致点が少ない領域を、現在のフレームへコピーする。逆に同一または類似ではない時には、画像バッファから現在のフレームにコピーする。このとき、動き予測部2以外の部分では、画像バッファの比較したブロックに相当するデータをエントロピー符号器12によって符号化する。

【0068】双方向予測では、同一または類似でないとは判別した時、動き予測部2以外では、画像バッファの比較したブロックに相当するデータをエントロピー符号器12によって符号化する。同一または類似であると判別した時は、順方向予測または逆方向予測と同一となる。

【0069】順、逆および双方向予測ともに、同一または類似であると判別した時の、最も不一致点が少ない領域と、現在判断している領域との相対位置のずれを、動きベクトルVとする。この動きベクトルVは、動きベクトルV用Vに決める符号化手順により圧縮や多重化を行う。

【0070】ここで、ブロックをN画素×N画素単位と

すると、このブロックの上位の $M^2$ （Mは2以上の整数）個の $N \times N$ 単位のブロックによって $MN \times MN$ のマクロブロックMBを構成し、マクロブロックMBごとに動きベクトルVを使用するか否かを判断してもよい。その場合には、ブロックごとの判断の閾値の他に、マクロブロックMBに対する閾値を定め、その閾値に対しての判断を行う方法の他に、ブロックとマクロブロックMBの双方の閾値を使う方法がある。

【0071】次に、この動き予測部2の具体的な動きを図を参照しながら説明する。従来と同様に、全く予測を用いないフレームをIピクチャー、過去の画像を用いて予測を行うフレームをPピクチャー、過去と未来の画像を利用して予測を行うフレームをBピクチャーとする。例えば図5（A）に示すように、入力順は、PBBPBBPBBPBBPBBとなる。このように、各フレームに役割を与えると、符号化順序は図5（B）に示すとおりとなる。すなわち、例えば、No. 8のフレームは、No. 7のIピクチャーとNo. 10のPピクチャーが符号化された後に、それらのデータを利用して符号化される。この符号化の際、動き補償が適宜使用される。

【0072】動き補償は、PピクチャーとBピクチャーを符号化するときに使用される。具体的には、符号化しようとするフレーム内を、図6に示すように、 $16 \times 16$ ピクセルのマクロブロックMBに分割し、そのマクロブロックMBと同一と判定されるマクロブロックを参照フレームから捜し出す。なお、参照フレームで、Pピクチャーを符号化するときはIピクチャーで、Bピクチャーを符号化するときはIピクチャーとPピクチャーとなる。

【0073】参照フレームから同一ブロックを捜し出す方法は、従来と同様に、参照フレームの対応するマクロブロックMBの位置およびその位置からわずかにずれた位置のマクロブロックMBの各画素を対比することにより行う。この実施の形態では、同一か否かの判断を両マクロブロック内の対応する位置の画素が異なる場合のその個数で行っている。すなわち、不一致数が所定値以下となるときに、動き補償部2の判断手段2aが両マクロブロックMBを同一と判断する。この実施の形態では、このマクロブロックMBの判定のための所定値を20個としている。

【0074】同一と判断すると、動き予測部2は、両マクロブロックMBの移動方向をベクトル化した動きベクトルVを生成し、その値をハフマンテーブルを利用した一方のエントロピー符号器12へ送出する。動き予測部2は、マクロブロックMBを4つに分割して小さな4つのブロックSBを生成する小ブロック化手段2bを有している。そして、この小さいなブロックSBについても同一か否かを判断手段2aにて判定している。この実施の形態では、そのブロックSBと参照フレーム中のブ

10

20

30

40

50

ロックSBの画素の不一致数が3個以下のとき、両ブロックSBは同一と判断している。同一か否かの判定基準となる所定数としては、0～5個の値とするのが、マルチカラー画像の乱れが生じつつ効果的な圧縮を行うためには好ましい。なお、ブロックSBはここではマクロブロックMBを4つに分割しているが、そのブロック数は4つに限るものではなく、また最も小さいブロックとしては、1画素も許される。

【0075】なお、この小さなブロックSBについては、動きベクトルV用のデータ部分を設けず、先のマクロブロックMBの動きベクトルVをそのまま利用する。このため、動きベクトルV用のデータ部分は小さくなり、データ増大を招くことがない。一方、同一か否かの判定は、小さなブロックSBで厳しく行うことができるので、マルチカラー画像の乱れも生じない。

【0076】例えば、図6に示すマクロブロックMBと、参照フレームのマクロブロックMBとの画素の不一致数が仮に18個とすると、両マクロブロックMBは同一と判定され、動きベクトルVが生成される。一方、小さなブロックSB(1)の不一致数が3個、SB(2)が2個、SB(3)が2個とすると、小さなブロックSB(1)、SB(2)、SB(3)は同一のものがあると判定され、マクロブロックMBによって生成された動きベクトルVを利用してデータ圧縮される。しかし、小さなブロックSB(4)の不一致数は11個となることとなり、この小さなブロックSB(4)は、動き補償を使用することなく、その画素が符号化される。

【0077】この実施の形態では、マクロブロックMBの同一判定を甘くし、小さなブロックSBの同一判定を厳しくしている。このため、動きベクトルを利用できるブロックが増える一方、画像の乱れも生じないものとなる。なお、画像の性質によっては、マクロブロックMBの同一判定を厳しくし、小さなブロックSBの同一判定を甘くしたり、両者を同一レベルのものとしても良い。

【0078】インデックス変換部7は、図2に示すように、参照画素生成手段となるラインバッファ10と、縮退手段となるマクロモデル生成部11と、エントロピー符号化手段となるエントロピー符号器12とに接続され、状態信号STを生成する状態生成部13と、プリスキャン時に2種類の交換テーブルを生成する交換テーブル生成部14と、参照順位テーブル15と、符号化時に画素インデックス(カラーシンボル)3を入力し、インデックス変換する画素インデックス変換テーブル16と、カラーパレット変換部17と、交換された変換カラーパレット18と、交換された画素インデックスを入力し、所定の順位を出力する判別部19とを含むように構成されている。

【0079】なお、この実施の形態において、符号化の対象となるカラー画素は、マルチカラーの画素インデックス100Aであり、1画素当たり8ビットのインデ

クスコードのデータで構成され、256色分のカラーシンボルを表示できる。

【0080】ラインバッファ10は、プリスキャン時には、参照画素を状態生成部13に入力し、符号化時には参照画素をマクロモデル生成部11に入力している。マクロモデル生成部11は、図7(A)に示すように、符号化対象画素buf[i]に対し参照画素として、周辺の4画素R[0]～R[3]を取り入れ、マクロ状態信号CXを発生させている。

【0081】エントロピー符号器12は、後述するように、予測ランレングス符号化によってモデル化されて画像信号を符号化コード化するものとなっている。このエントロピー符号器12は、図40に示す算術型のエントロピー符号器や、ハフマン符号を使用した符号器を採用するようにしても良い。なお、エントロピー符号器12の構成、動作、機能の詳細については後述する。

【0082】状態生成部13は、符号化プロセスに先立つプリスキャン時に動作するもので、図7(A)に示すように、符号化対象画素をbuf[i]としたとき、その周辺の参照画素R[0]～R[3]の状態を図8に示すように区別するものとなっている。例えば、R[0]～R[3]がすべて同一の色の時は、状態信号STは「0」となり、R[0]～R[3]がすべて互いに異なる色のときは状態信号STは「33」となる。なお、符号化対象画素buf[i]は、先頭ラスタにあるときは、図7(B)に示すように、現れていない部分である参照画素R[1]～R[3]にはすべて「0」、すなわち後述するように最も頻度の高い画素インデックスを設定する。また、符号化対象画素buf[i]が、画像の先頭であるとき、R[0]～R[3]のすべてに「0」を設定する。

【0083】交換テーブル生成部14も、プリスキャン時にのみ動作する。そして、符号化対象画素buf[i]が4つの参照画素(以下R[4]という)と一致した場合、状態信号毎にその一致した色の位置の度数を算出すると共に、参照画素R[4]の中に符号化対象画素buf[i]が無いときには、その一致しなかった画素インデックスの出現度数をカウントする。

【0084】例えば、状態信号STが「3」のとき、図9(A)に示すように、符号化対象画素buf[i]が参照画素R[0]と一致した数をNAとし、参照画素R[2]と一致した数をNBとすると、NA>NBならば作成する参照順位テーブル15の参照画素位置は、図9(B)に示すように、0位にR[0]がきて、1位にR[2]がくる。一方、NA<NBのときは、図9(C)に示すように、0位にR[2]、1位にR[0]がくる。このようにして、図10に示すような参照順位テーブル15が作成される。なお、参照順位テーブル15中の数字、例えば、状態信号STが「3」のときの0位の「2」、1位の「0」は、先に示した例で言えば、0位

のR〔2〕の「2」を、1位のR〔0〕の「0」を示している。また、状態信号STが「0」のときは0位のみのであり、参照順位テーブル15から除かれるため、そのテーブル15は縦が14個、横が4個のテーブルとなっている。

【0085】また、変換テーブル生成部14は、参照画素R〔4〕の中に符号化対象画素buf〔i〕の色が無いときは、図11に示すように、参照画素R〔4〕中になかった度数を画素インデックス毎にカウントする。そして、図12に示すような画素インデックス変換テーブル16を生成する。なお、図12に示すものは、 $N2 \geq N0 \geq \dots \geq Nn$ の場合となっている。

【0086】カラーパレット変換部17は、カラーパレット（例えば、図13の左側に示すようなカラーパレット）9を、画素インデックス変換テーブル16を利用して図13（B）に示すような変換カラーパレット18を生成している。例えば、画素インデックス（カラーシンボル）C2がRGBの三原色の組み合わせとしてRGB2なるものとされているとき、参照画素R〔4〕中に無かった度数の順位が1番目すなわち順位として0順位のとき、変換カラーパレット18では最上位すなわちカラーシンボルC0の位置に設定されることとなる。

【0087】判別部19は、符号化時に動作するもので、画素インデックス変換テーブル16によって変換された画素インデックスと、状態信号STに基づく参照画素が入力される。そして、その画素インデックスと参照画素が一致したときは、参照順位テーブル15に基づいてその順位を出力する。一致しなかった場合は、その画素インデックスの順位に、マルコフ状態信号CXに1を加えた数を加算して出力する。例えば、変換された画素インデックスがCnのときで状態信号STが「3」の場合は、その画素インデックスCnと同一の色が参照画素R〔0〕にあったときは、順位として「1」を出力する。一方、画素インデックスCnが参照画素R〔4〕中に無いときは、CXが「1」なので、「 $n+1+1$ 」=「 $n+2$ 」を出力する。

【0088】エントロピー符号器12は、判別部19の出力をマルコフ状態信号CXでマルコフモデル化し、可変長符号化する。合成部20では、変換カラーパレット18のデータと、参照順位テーブル15のデータと、エントロピー符号器12からの出力とを合成して符号化データ200として出力する。この実施の形態では、符号化データ200は、画像のサイズ等の情報に続き、変換カラーパレット18のデータ、参照順位テーブル15のデータ、最後にエントロピー符号器12によって符号化された符号化コードという順に出力されるが、基本的には符号化データ200中に入っていれば良く、他の順序としても良い。

【0089】次に、以上の構成を有する符号化システム1の動作について、図14および図15に示すフローチ

ャートに基づいて説明する。なお、動き予測部2の動作は、既に説明したので、ここではインデックス変換部7およびその周辺の部材の動作について説明する。

【0090】符号化システム1の動作は、プリスキャンプロセスと符号化プロセスとに分かれる。符号化の前処理に当たるプリスキャンプロセスでは、後段の符号化プロセスで適切な符号化が行えるように、画素インデックス変換テーブル16および参照順位テーブル15の作成を行う。このために、まず、スイッチ21を端子21aに接続する。そして、ステップS0では変換テーブル生成部14中の後述するカウントテーブルならびに参照順位テーブル15および画素インデックス変換テーブル16の初期化ならびに後述する符号化処理のための各種変数の初期化を行う。

【0091】ステップS1では、入力画像のカラーパレット9のデータをカラーパレット変換部17中の配列palette〔 〕に入力する。ステップS2では、入力画像の画素インデックス100Aをラインバッファ10中の配列buf〔 〕に入力する。次に、ステップS3では、符号化対象画素buf〔i〕の周辺の1画素をラインバッファ10から取り出し、変換テーブル生成部14や状態生成部13等の配列R〔 〕に入力する。符号化対象画素buf〔i〕と参照画素R〔0〕～R〔3〕の位置関係は図7に示すとおりである。ステップS4では、ラインバッファ10から抽出した参照画素に基づき、図8の基準で状態信号STを生成する。

【0092】ステップS5では、参照画素R〔4〕の中に符号化対象画素buf〔i〕と一致するものがあるかどうか調べる。その後、ステップ6で参照画素R〔i〕と一致した場合、変換テーブル生成部14中の参照順位テーブル15用のカウントテーブルN\_table\_A〔ST+j〕に1を加える。ここで、jは参照画素の位置を現すため、N\_table\_A〔 〕には、参照画素の状態と一致した位置毎に、一致した回数がカウントされる。次のステップS7では、参照画素と一致しない場合、変換テーブル生成部14中の画素インデックス変換テーブル16作成用のカウントテーブルN\_table\_B〔buf〔i〕〕に1を加える。つまり、カウントテーブルN\_table\_B〔buf〔i〕〕には、参照画素と一致しなかった画素について、その画素インデックスごとにカウントされる。

【0093】ステップS8で、すべての画像データについて、ステップS3からステップ6あるいはステップ7までの処理を繰り返す。カウントテーブルN\_table\_B〔 〕には、画素インデックス100A毎に出現度数がカウントされている。ステップS9では、これを基に出現度数の多いものから昇順のコードに変換する画素インデックス変換テーブル16を作成する。このテーブル16により、出現度数の多い画素インデックス100Aほど、小さいインデックス値に変換され、その特性

に合わせたエントロピー符号化を用意すれば、いかなる画像であっても最も効率的な良いエントロピー圧縮を行うことができる。

【0094】ステップS10では、その参照順位を記述した参照順位テーブル15を作成する。すなわち、カウントテーブルN\_table\_A〔 〕には、参照画像の状態信号STと、符号化対象画像buf〔i〕に一致した参照画像位置毎にその度数がカウントされているので、度数の多い順に昇順を付ければ、最も効率的な良い圧縮が期待できる。

【0095】プリスキャンプロセスによって、参照順位テーブル15と画像インデックス変換テーブル16を作成後、図15に示す符号化プロセスを実行する。この実行に当たり、まずスイッチ2を端子211側に切り換える。

【0096】プリスキャンプロセスにて作成された参照順位テーブル15は、復号処理に必要となる。このため、圧縮ストリームへのヘッダ情報の一部として符号化データ200のストリームに付加するために、ステップS11で、合成部20に出力される。ステップS12では、プリスキャンプロセスで作成した画像インデックス変換テーブル16を用いて、画像インデックス100Aの変換を行う。この変換により参照画像と一致しなかった画像インデックス100Aが、その出現度数の大きい順に昇順のコードとなる。

【0097】画像インデックス100Aの変換に伴って、カラーパレットも復号側で正しい色が再現されるように変換する必要がある。このため、ステップS13で、画像インデックス変換テーブル16を利用して、カラーパレット変換部17で変換カラーパレット18の作成を行う。変換された変換カラーパレット18は、合成部20に入力し、符号化データ200からなる圧縮ストリームのヘッダ情報の一部として付加される。ステップS14では、プリスキャンプロセスのステップS3と同じ動作をする。すなわち、ラインバッファ10から参照画像R〔4〕を取り出し、マルコフモデル生成部11や状態生成部13に入力する。

【0098】次のステップS15では、プリスキャンプロセスのステップS4と同じ動作をする。すなわち、状態生成部13で、図8に示す基準に基づいて状態信号STを、参照順位テーブル15に向けて出力する。ステップS16では、マルコフ状態信号CXを生成する。このマルコフ状態信号CXは、参照画像の色数を見す値で、図8の表に基づいて参照画像の状態から生成する。なお、この符号化プロセスのフローでは明記しないが、このマルコフ状態信号CXは、エントロピー符号化を行う際のマルコフモデル化のための符号化条件としても用いている。

【0099】ステップS17では、符号化対象画像buf〔i〕が、参照画像R〔4〕と一致するか否か調べ

る。具体的には、参照順位テーブル15中の状態信号STに対応する欄内を0からCXと等しい値まで変化させ、符号化対象画像buf〔i〕と一致するか調べていく。参照画像と一致した場合は、即ステップS18に進み、そうでない場合はステップS19に進む。

【0100】参照画像と一致した場合は、その時の順位をエントロピー符号化する(ステップS18)。参照画像と一致しなかった場合は、符号化対象画像buf〔i〕を直接エントロピー符号化する。ただし、(CX+1)個の符号が参照画像と一致した場合の符号に割り当てられているので、(CX+1)を加算して符号化する(ステップS19)。ステップ20で、すべての画像データについて、ステップS14からステップS18あるいはステップS19までの処理を繰り返す。

【0101】ところで、上述したフローでは、ステップS18あるいはステップS19で順位の確定と共に逐次エントロピー符号化を行うようになっているが、全画像の順位と対応するマルコフ状態信号CXを一旦バッファに格納し、まとめて後述する符号化を行っても良い。例えば、後述する予測ランゲルスの符号化方法を採用した場合、符号化対象シンボル以降の未来のシンボルをまとめて符号化することとなる。

【0102】なお、PピクチャーやBピクチャーの符号化において、所定のマクロブロックMBが動き補償を行わないと判断されたとき、それらのマクロブロックMBが拾われ、連続して符号化される。例えば、図16

(A)に示すように、マクロブロックMBのNo. 1〜No. 7が動き補償を行わないものとなるとき、符号化はNo. 1→No. 4→No. 2→No. 6→No. 7→No. 5→No. 3の順で行われる。このため、メモリ容量を小さくしても効果的な圧縮が可能となる。動き予測の結果、動きベクトルVを利用しないで、エントロピー符号化を行い復号化システム3に伝送する場合、マクロブロックMBの符号化の再には、図16(B)のようにマクロブロックMB内をラスタスキャンによって画素値をラインバッファ10に格納しつつ符号化する。例えば、図16(B)に示す、No. 1, 4, 2のマクロブロックMBをNo. 1→No. 2の順で符号化すると、No. 2のマクロブロックMBのNo. 1と隣接する部分の画素を効果的に圧縮するには、No. 1のマクロブロックMBの右端側の少なくとも1列について記憶するためのメモリが余分に必要となると共にアクセスの手間がかかる。ところが、No. 1→No. 4の順で符号化すると、No. 4のマクロブロックMBのNo. 1と隣接する部分の画素の符号化には先に示した最新出現表をそのまま利用でき、余分のメモリが必要とならずアクセス時間も速くなる。

【0103】なお、復号化システム3に伝送されるブロックは、1度エントロピー符号化をかけられるが、動きベクトルVを使用することによって画像そのものを復号

10

20

30

40

50

化システム3に伝送しなくなるブロックは1度もエントロピー符号化がかけられないこととなる。

【0104】次に、符号化システム1に対応したマルチカラー画像からなる動画の復号化システム3について説明する。

【0105】この復号化システム3は、動き補償部4と、後述するインデックス変換部8からのデータと動き補償部4からのデータとを加算する加算器3aと、2つのフレームメモリ3b、3cと、各スイッチ6a、6b、6c、6dを切り替える制御回路6と、インデックス変換部8と、周辺画素生成手段および参照画素生成手段となるラインバッファ30と、縮退手段となるマルコフモデル生成部31と、エントロピー復号化手段となる2つのエントロピー復号器32、32aと、分離部33とを含み、入力される符号化データ200のデータストリームをカラー画素データ100Bのデータストリームに変換し出力するように形成されている。

【0106】このとき、復号化システム3のアルゴリズムは、符号化システム1のアルゴリズムと全く逆のアルゴリズムになるように構成されている。したがってカラー画素データ100Aのビット構成およびデータストリームと、カラー画素データ100Bのビット構成およびデータストリームとは全く同じものとなる。また、動き補償部4は、先の符号化システム1の動き予測部2と基本的に同一となっており、その内部に、動きベクトルVの有無を判断する判断手段4aを有している。

【0107】すなわち、動き補償部4の動作は、動き予測部2と逆の動作を行う。動きベクトルVの情報を元に、動きベクトルVと、その動きベクトルVを使用するか否かの情報を入力する。動きベクトルVを使用する場合には、フレームメモリ3b、3c中のデータを復号し、動きベクトルVを使用しない場合には、エントロピー復号器32から復号したデータを、復号データとして採用する。

【0108】動きベクトルVを使用する場合、順方向のときには過去のデータから、逆方向のときには未来のデータから、それぞれデータを転送する。双方向の場合には、過去または未来のいずれかを選択し転送する。一方、動きベクトルVを使用しない場合には、エントロピー復号器32からのデータを復号データとして出力する他に、順、逆方向の場合には、現在のフレームデータとしてフレームメモリ3b、3cに格納する。

【0109】フレーム間予測符号の差分符号化に関しては、復号化システム3には加算器3aを含むこととなる。符号化システム1の構成に準じて、加算器3aを迂回させたりまたは、加算器3aのもう一方の入力に0を入力することによってフレーム間予測符号の差分符号化を回避させることができる。

【0110】インデックス変換部8は、参照画素生成手段となるラインバッファ30と、縮退手段となるマルコ

フモデル生成部31と、エントロピー復号化手段となるエントロピー復号器32とに接続され、状態信号STを生成する状態生成部34と、参照順位テーブル35と、画素インデックス（カラーシンボル）生成部36とを含むように構成されている。

【0111】ラインバッファ30は、符号化システム1のラインバッファ10と同様に複数の画素を保存できるものとなっている。そして、その値を周辺画素としてマルコフモデル作成部31や状態生成部34等に入力し、参照順位テーブル35や各信号ST、CXを作成している。

【0112】エントロピー復号器32は、マルコフ状態信号CXを利用して、入力する符号化データ200中の符号化コードを、エントロピー符号器12と逆の手順で復号化演算処理する。なお、エントロピー復号器32は、エントロピー符号器12とは全く逆のアルゴリズムでその演算を行うように形成しなければならない。したがって、エントロピー復号器32は、エントロピー符号器12に算術符号器が用いられた場合には、それと同様な構成の算術復号器として形成する必要がある。また、エントロピー符号器12にハフマン符号器が用いられた場合には、それと同じ構成のハフマン復号器として構成する必要がある。

【0113】分離部33は、符号化データ200のデータストリームより変換カラーパレット18のデータと、参照順位テーブル35のデータとを分離するものである。そして、その内部に符号化システム1の合成部20と逆の機能が設定されており、エントロピー符号器12で符号化された符号化コードをエントロピー復号器32に出力する。

【0114】状態生成部34は、符号化システム1中の状態生成部13と同様な構成、機能を有しており、図8に示す表に示された基準に基づいて状態信号STを出力する。参照順位テーブル35は、分離部33で分離された参照順位テーブル35用のデータによって構成され、符号化システム1中の参照順位テーブル15と同様な表となっている。画素インデックス生成部36は、符号化システム1で生成された所定順位のデータおよびマルコフ状態信号CXならびに参照順位テーブル35の各データから画素インデックス6を出力する。

【0115】次に、このように構成される復号化システム3の動作について、図17に示すフローチャートに基づいて説明する。

【0116】まず、ステップS30で、参照画素や後述する予測ランレングス（PRLC）復号のための各種変数の初期化を行う。次に、分離部33によって符号化データ200の圧縮ストリームのヘッダに付加された参照順位テーブル35のデータを抽出し参照順位テーブル35を生成する（ステップS31）。ステップS32では、圧縮ストリームとなる符号化データ200のヘッダ



に付加された変換カラーバレット18のデータを同様に分離部33によって抽出し出力する。

【0117】次に、ステップS33で、参照画像R

【4】を抽出する。これは符号化プロセスのステップS14と全く同様となっている。ステップS34では、マルコフ状態信号CXを生成する。これは符号化プロセスのステップS16と全く同様となっている。ステップS35では、PRLCによるエントロピー復号により復号順位Xを得る。

【0118】ステップS36では、画像インデックス生成部36において、順位Xがマルコフ状態信号CXの値以下かどうかを判定する。順位Xが値CX以下の場合、復号される画像インデックスが参照画像に存在することを示し、ステップS37に進む。それ以外では、ステップS38に進む。ステップS37では、状態信号STで区分される中に存在する順位Xが指し示す位置にある参照画像の画像インデックス100Bを出力する。

【0119】ステップS38では、復号順位Xが参照画像を指し示さない場合は、復号順位Xそのものが復号される画像インデックス6となる。ただし、 $(CX+1)$ の値の符号が参照画像に割り当てられているため、復号順位Xから $(CX+1)$ を差し引く必要がある。ステップS39で、すべての画像インデックス6の復号が完了するまで、ステップS33からステップS37あるいはステップS38までを繰り返す。そして、すべての画像インデックス6の復号が終了すると、復号動作が終了する。

【0120】次に、先に説明した符号化システム1中のエントロピー符号器12で行う符号化について説明する。この実施の形態では、いわば予測ランレングス（以下、PRLCという）と称するものを採用している。このPRLCは、良く知られているランレングス符号化を改良したもので、そのアルゴリズムの概要について、図18から図20に基づいて説明すると共に、このPRLCの基礎となる基本的な符号化方法等を図21および図22に基づいて説明する。

【0121】この符号化の基本のアルゴリズムは、従来から良く知られているQMコードと同様、基本的には、2値のビット列を圧縮の対象としている。まず初期値として、“0”か“1”のいずれかを優勢シンボルと定め、そのシンボルが連続すると予測する個数runを設定する。入力系列の出現確率が不明の場合は、runを1に設定するのが良い。その上で、以下に示すようなルールに従い符号化を進める。なお、個数runが予測ビット数に相当する。

【0122】図18に示すように、runで示される注目系列がすべて優勢シンボルであると予測し、予測が当たったとき、符号語として“0”を出力し、この系列の符号化を完了する。はずれた場合は“1”を出力し、次の分割符号化工程を実行する。

【0123】予測がはずれた場合は、図19に示すように注目系列を前半部系列と後半部系列の2つに分け、前半部がすべて優勢シンボルのときは符号語として“0”を、前半部系列に劣勢シンボルが存在し後半部がすべて優勢シンボルのときは、符号語として“10”を、前半部にも後半部にも劣勢シンボルが存在するときは“11”をそれぞれ出力する。そして、劣勢シンボルが存在する系列は、さらに系列を分割する（図20参照）。劣勢シンボルが存在する系列は、可能な限り系列を分割し上述の分割符号化工程を繰り返す。

【0124】なお、分割は必ずしも2つの均等分割とする必要はなく、不均等な分割としたり3つ以上の分割としても良い。また、予測が当たったとき“0”ではなく、優勢シンボルを出力し、はずれた場合“1”ではなく、劣勢シンボルを出力するようにしたり、予測当たりで“1”を、予測はずれで“0”を出力するようにしても良い。

【0125】以上がこの符号化の前提となるデータ符号化の基本アルゴリズムであるが、さらに、入力系列の出現確率の変化に追随し、符号化効率を向上させるため、以下の処理を加えるようにしても良い。

【0126】すなわち、runで予測した系列が続いて所定回数、例えば、2回当たったとき、runを2倍等に増加させる。なお、予測が的中し続けた場合、さらに予測範囲を拡大していく。一方、runで予測した系列が2回以上劣勢シンボルを含むと共にrunを1/2倍する。予測がはずれ続けた場合、さらに1/2倍していく。そして、runが1で、それが劣勢シンボルのときは、以降の入力系列を反転させる。すなわち、優勢シンボルを変更させる。

【0127】なお、runで予測した系列の後半部系列に劣勢シンボルが存在するとき、runをいきなり1/4等に減少させるようにしても良い。これは、後半部に劣勢シンボルが存在するときは、次に続く系列に劣勢シンボルが多く含まれると判断されているためである。このため、runで予測した系列の前半部系列のみに劣勢シンボルが存在するとき、後半部に劣勢シンボルが存在するときより多い値、例えばrunを1/2倍するようにしても良い。

【0128】この実施の形態のエントロピー符号器12内の符号化プロセスは、次に説明する図21および図22の符号化プロセスを改良したものであり、まずその符号化プロセスについて説明する。改良前の符号化プロセスは、図21の符号化メインルーチンと図22の符号化サブルーチンにより構成される。なお、図22中の符号化サブルーチンは、サブルーチンから同じサブルーチン呼び出すいわゆる関数の再帰読出しを行っている。

【0129】まず、図21の符号化メインルーチンの各ステップについて説明する。なお、符号化の対象は2値のビット列からなる入力系列となっている。最初に、予

測の初期値  $run$  の設定と優勢シンボルの選択 ("0" または "1") を行う (ステップ S50)。次に、ローカル変数  $ofs$  に 0 を、 $width$  に  $run$  を代入する (ステップ S51)。ここで  $ofs$  は、符号化のために予め定義した配列  $A$  のポインタで、予測開始ビット位置を示す。したがって初期値は 0 となる。 $width$  は  $ofs$  で示したビット位置から何ビットを予測の対象にするかを示す値で、ここでは、予測の初期値  $run$  が代入される。その後、予め定義した配列  $A$  の  $A[ofs]$  から  $A[width-1]$  までに入力ビットを書き込む (ステップ S52)。そして、 $A[ofs]$  から  $A[width-1]$  のすべての要素が優勢シンボルのときステップ S54 へ進み、ひとつでも劣勢シンボルが含まれているときは、ステップ S55 へ進む。

【0130】予測が的中した場合、符号語として予測当たり信号 "0" を出力し、配列  $A$  に取り込んだ系列の符号化を完了する (ステップ S54)。一方、予測ははずれた場合、符号語として予測はずれ信号 "1" を出力する (ステップ S55)。そして、 $width$  が 1 以上か否かを検出する (ステップ S56)。 $width$  が 1 以下ならこれ以上分割できないので、ステップ S7 の符号化サブルーチンへは移行せずステップ S58 へ移行する。一方、 $width$  が 1 を超えていると、図 22 の符号化サブルーチン呼び出し (ステップ S57)。

【0131】ステップ S58 では、予測  $run$  の再設定と必要ならば優勢シンボルの変更を行う。すなわち、このステップ S58 においては、基本的に予測が的中すれば、 $run$  を大きくし、はずれれば小さくする。そして  $run$  を小さくしても予測が所定回数はずれ続けるようなら、優勢シンボルの変更を行う。なお、予測の的中や予測のはずれをどのように評価するかについては、さまざまな方法を採用することができる。たとえば、予測がはずれた場合、直ちに  $run$  を小さくしたり、2 回以上連続してはずれたとき、初めて  $run$  を小さくする等の方法を採用することができる。さらに、前半部系列もしくは後半部系列のみはずれた場合と、両方はずれた場合とで  $run$  の縮小の度合いを異ならせる方法も採用できる。また、符号済みビット系列で所定の確率テーブルを引き、次の予測  $run$  を設定する等の方式も採用可能である。

【0132】符号化メインルーチンで 1 次予測がはずれた場合は、ステップ S57 で図 22 に示す符号化サブルーチン呼び出し。符号化サブルーチンへ渡す引き数は、 $ofs$  と  $width$  である。以下、符号化サブルーチンの各ステップについて説明する。

【0133】符号化サブルーチンでは、予測を前半部系列と後半部系列に分けて行うため、予測の範囲を半分にする (ステップ S60)。すなわち、親ルーチンから引き数として受け取った  $width$  を  $1/2$  にする。そして、次のステップ S61 で、前半部系列 (配列  $A[ofs]$

$fs]$  から  $A[ofs+width-1]$  まで) がすべて優勢シンボルか否かをチェックする。すべて優勢シンボルならステップ S62 へ進む。ひとつでも劣勢シンボルが存在したら、直ちにステップ S64 へ進む。

【0134】前半部系列がすべて優勢シンボルなら、符号語として "0" を出力する (ステップ S62)。そして、前半部系列の先頭位置を示すポインタ  $ofs$  に  $width$  を加え、後半部系列の先頭位置を示すように変更する。また、前半部系列がすべて優勢シンボルのときは、後半部系列に必ず劣勢シンボルが存在するので、後半部系列の予測がはずれたことを示す符号語 "1" を出力する必要がない。したがって、後述するステップ S70 はスキップし、ステップ S71 に進む。

【0135】一方、前半部系列に劣勢シンボルが存在する場合、符号語として "1" を出力する (ステップ S64)。次に、 $width$  が 1 を超えているか否かをチェックする (ステップ S65)。1 以下の場合、これ以上分割できないので、子の符号化サブルーチン (ステップ S66) の呼び出しをスキップし、ステップ S67 へ移行する。なお、 $width$  が 2 以上なら、さらに系列を 2 つに分け、それぞれを符号化しなければならぬ。そのための子の符号化サブルーチン呼び出し (ステップ S66)。子の符号化サブルーチンは、図 22 に示した符号化サブルーチンと全く同一となっている。つまり、ここでは、同一ルーチン (関数) の再帰呼び出しを行う。

【0136】符号化サブルーチンの再帰呼び出しによって前半部系列の符号化を終了すると、前半部系列の先頭位置を示すポインタ  $ofs$  にステップ S60 で設定した  $width$  を加え、後半部系列の先頭位置を示すように変更する (ステップ S67)。その後、後半部系列 (配列  $A[ofs]$  から  $A[ofs+width-1]$  まで) がすべて優勢シンボルか否かをチェックする (ステップ S68)。すべて優勢シンボルならステップ S69 へ進む。ひとつでも劣勢シンボルが存在したら、直ちにステップ S70 へ進む。そして、後半部系列がすべて優勢シンボルなら、符号語として "0" を出力する (ステップ S69)。

【0137】一方、前半部系列に劣勢シンボルが存在する場合、符号語として "1" を出力する (ステップ S70)。そして、次に、 $width$  が 1 を超えているか否かをチェックする (ステップ S71)。1 以下の場合、これ以上分割できないので、子の符号化サブルーチンを実行するステップ S72 をスキップし、次の注目系列の符号化工程へリターンする。なお、後半部系列についても、 $width$  が 2 以上なら、さらに系列を 2 つに分け、それぞれ符号化する。そのため図 22 に示す符号化サブルーチンと同一の子の符号化サブルーチン呼び出し (ステップ S72)。この符号化サブルーチンの再帰呼び出しによって後半部系列の符号化を実行する。

【0138】以上のような符号化プロセスの具体例を次に説明する。すなわち、符号化の具体例として、予測の初期値  $run$  を 8、優勢シンボルを “0” として、“0001001” として表される入力ビットを符号化する場合について説明する。

【0139】まず、図 21 の符号化メインルーチンのステップ S52 で、A [0] から A [7] に、上記の入力ビットを入力する。ステップ S53 では、A [0] から A [7] のすべてが “0” かどうか判定する。上の例の場合、ビット列に “1” が含まれているので、ステップ S55 に移行し、まず符号語として “1” を出力する。続いてステップ S56 では、 $width$  の大きさをチェックするが、 $width$  はこのとき 8 なので、符号化サブルーチン (ステップ S57) に進む。

【0140】符号化サブルーチンでは、まずステップ S60 で、 $width$  を  $1/2$  の 4 に設定する。そしてステップ S61 で、入力ビットの前半部、つまり A [0] から A [3] がすべて 0 かどうかチェックする。この場合、すべて “0” なのでステップ S62 に進み、符号語として “0” を出力する。以上で前半部系列の符号化が完了する。続いてステップ S63 を実行し、後半部系列の符号化に移るが、前半部系列がすべて “0” の場合、後半部系列に “1” が含まれるのは明らかである。したがって、ステップ S71 で  $width$  が 1 以下でない限り後半部系列をさらに分割して符号化しなければならない。そこで、符号化サブルーチンをサブプロセスとしてステップ S72 で再び呼び出す。なお、そのための前処理として、上述したようにステップ S63 では、 $ofs$  に  $width$  を加え、 $ofs$  を後半部系列の先頭位置にセットする。

【0141】ステップ S72 では、 $ofs$  と  $width$  を引き数として子の符号化サブルーチンを呼び出す。子の符号化サブルーチンを実行するステップ S72 では、まず、図 22 に示す符号化サブルーチンのステップ S60 で  $width$  をさらに半分にして 2 に変更する。次のステップ S61 では、前半部系列、すなわち A [4] と A [5] が共に “0” であるか否かをチェックする。この場合、A [4] が “1” なので、次のステップ S64 に移行し、符号語として “1” を出力する。そしてステップ S65 で  $width$  が 1 を超えていると判断し、孫プロセスをステップ S66 で呼び出す。孫の符号化サブルーチンでは、まずステップ S60 において  $width$  が 1 となる。A [4] は “1” なのでステップ S61 からステップ S64 へ処理が移り、符号語 “1” を出力する。ステップ S65 では、 $width$  が 1 以下なので、ステップ S66 をスキップし、ステップ S67 で  $ofs$  を 5 に変更する。A [5] は “0” なのでステップ S68 からステップ S69 に処理が移り、符号語 “0” を出力する。

【0142】次に、この孫の符号化サブルーチンから抜

けて、子の符号化サブルーチンのステップ S67 に戻る。子の符号化サブルーチンの  $ofs$  は 4、 $width$  は 2 であるから、ステップ S67 で  $ofs$  は 6 に変更される。したがってステップ S68 では、A [6] と A [7] をチェックすることになる。この場合、A [7] が “1” なのでステップ S70 へ移行し、符号語 “1” を出力する。そして、再び孫の符号化サブルーチンをステップ S72 で呼び出す。孫の符号化サブルーチンでは、A [6] が “0” なのでステップ S62 で符号語 “0” を出力する。そして、 $width$  が 1 なので、ステップ S72 をスキップして子の符号化サブルーチンに復帰する。

【0143】子の符号化サブルーチンに復帰したプロセスは、さらに符号化メインルーチンに復帰し、ステップ S58 で予測  $run$  の再設定と、優勢シンボルの再設定を行う。この例の場合、1 次予測ははずれたが、2 次予測で前半部が的中したので、 $run$  を 8 から 4 に変更し、優勢シンボルは引き続き “0” とする処理を施す。なお、予測  $run$  の設定は、2 回続けてはずれたときに変更する等の設定にしても良い。

【0144】このような符号化プロセスによって、入力ビットである “00001001” が “1011010” の符号化系列となる。したがってこの場合、8 ビットの入力系列が 7 ビットに圧縮されたことになる。

【0145】なお、復号化プロセスについては、符号化プロセスと逆のアルゴリズムによって、入力されてくる符号語を復号している。すなわち、復号化プロセスも、復号化メインルーチンと復号化サブルーチンにより構成され、符号化と逆のアルゴリズムによって復号している。

【0146】このように、図 21 および図 22 に示す符号化プロセスおよびその符号化プロセスと逆のアルゴリズムを使用して行う復号化プロセスでは、圧縮率が従来の QM コーダと呼ばれるものと同レベルであり、一方、符号化時間や復号化時間は大幅に短縮されたものとなっている。しかし、この符号化プロセスおよび復号化プロセスにおいては、予測ビット数である  $run$  で定まるビット列を一度に符号化しているため、圧縮率を高めるために、 $run$  の最大値を大きく設定すると、 $run$  の個数分のビットを保持しておくためのバッファが大きくなるという問題が生じる。

【0147】この問題は、マルコフ状態信号  $CX$  をマルコフモデル化から得るような場合、そのバッファが非常に大きくなり、さらに大きな問題となる。すなわち、仮に  $run$  を  $n$  としたとき、バッファとしては  $n$  ビット分必要となり、さらに  $m$  状態のマルコフモデル化を行うと、バッファは各状態毎に必要となるため、 $n \times m$  ビットの容量になる。この容量は、 $run$  の値が大きくなると無視できなくなる大きさとなる。

【0148】また、図 21 および図 22 に示す符号化バ

ロセスおよびその符号化プロセスと逆のアルゴリズムを使用する復号化プロセスでは、その各処理時間は、QMコダに比べ大幅に短縮されているものの、符号化や復号化のサブルーチンを再帰的に呼び出して符号化や復号化を行っており、このサブルーチンの再帰的呼び出しのプロセスで時間を有するものとなっている。

【0149】このため、本実施の形態では、図18から図22に示す符号化プロセスおよび復号化プロセスを生かしつつ、デコード用のバッファを小さくしたり、符号化や復号化の時間をさらに減少できるデータ符号化方法等を採用している。以下、本発明の実施の形態におけるインデックス変換部7、8の動作の基本的な考え方を、図23から図34に基づき説明する。

【0150】まず、改良された本発明の実施の形態のエントロピー符号器12を、図23に基づき説明する。

【0151】このエントロピー符号器12は、エントロピー符号化装置となっており、符号化すべき2値ビット列を入力するビット列分解部22と、各予測ビット長run毎に符号化テーブルを内蔵する符号化テーブル部23と、符号化テーブル部23から入力される可変長符号を一旦バッファリングして固定のビット幅にならして出力するストリーム生成部24と、後述する状態遷移表を内蔵し、予測ビット長run等を設定する符号化制御部25とから主に構成される。

【0152】ここで、符号化制御部25には、状態遷移表を有する状態遷移部26と、マルコフモデル等により生成される符号化条件を入力し、その条件毎に現在の状態の信号を状態遷移部26に与え、符号化後に次の状態の信号を入力し、その符号化の状態を記憶しておく状態記憶部27とが設けられている。

【0153】したがって、マルコフモデルのような条件付き符号化を行うときは、条件をインデックスとして、状態記憶部27から該当する状態を取り出し、その状態を後述する図28に示した状態遷移表により遷移させ、次の状態を再び状態記憶部27の元の番地にストアしておけば、条件毎に、状態を管理できることとなる。したがって、予測ビット長run等のパラメータも条件毎に個別に設定できることとなる。なお、マルコフモデル化する場合、ビット列分解部22には、各符号化条件毎に切り換えるバッファが複数必要となるが、この実施の形態では、予測ビット長runの数ではなく、より小さい固定の区切りビット数pで段階的に符号化しているもので、そのバッファの容量はそれ程大きくならず、実用面で適したものとなっている。

【0154】ビット列分解部22は、符号化制御部25から予測ビット長runを指示する信号RUNと、優勢シンボルを指示する信号SWを入力する。ここで、信号RUNは、1からn(nは最大予測ビット長)の値を取る。また、信号SWは、その値が「0」のとき、「0」を優勢シンボルとし、「1」のとき、「1」を優勢シン

ボルとするが、その逆でも構わない。

【0155】さらに、ビット列分解部22は、デコードすべきビット数の信号DECNUMと、デコードすべきビットのパターンとなる信号パターンDECPATNを符号化テーブル部23に出力する。信号DECNUMは、入力ビット列に、優勢シンボルを含む4ビットのパターンが現れたとき、その4ビットとそれまで続いた優勢シンボル個数の合計数となる。なお、信号RUNが「4」未満のときは、信号RUNと同じ値が出力される。これは、この実施の形態では、区切りビット数pを「4」としているためである。

【0156】このようにして、ビット列分解部22は、入力したビット列が信号RUNで指定されたビット数分、すべて信号SWで指定された優勢シンボルが続いたとき、すなわち、予測的中したとき、信号DECNUMとして信号RUNの値を、信号パターンDECPATNとして「0」を出力する。

【0157】符号化テーブル部23は、図24から図27に示すような符号化テーブルを内蔵しており、どのテーブルを用いるかは、符号化制御部25からのテーブル番号指示信号TABLEにより選択される。そして、この符号化テーブル部23は、ビット列分解部22からの信号DECNUMと信号パターンDECPATNにより所定のテーブル内を検索し、所定の圧縮ビット列DECBITとそのビット長LENGTHおよび予測の当たり外れを示すFAILを出力する。なお、信号TABLEは、信号RUNと1対1の関係性を有するものとなっている。

【0158】図24の符号化テーブルは、テーブル番号が「0」で、信号RUNの値が「1」、すなわちrunが「1」の場合を示している。図24に示されるように、runが「1」のときは、2種類の信号となっている。すなわち、デコードすべきビット数は1個であり、信号パターンは「0」と「1」の2種類となる。この2種類の入力信号に対して、圧縮ビット列DECBITと、そのビット長LENGTHと、予測の内外れを示すフラグFAILの組み合わせからなる2種類の信号が対応する。例えば、信号DECNUMが「1」で、信号パターンDECPATNが「0」の場合は、予測当たりとなり、フラグFAILは当たり信号の「0」となり、圧縮ビット列DECBITは「0」となり、ビット長LENGTHは「1」となる。

【0159】図25の符号化テーブルは、テーブル番号が「1」で、runが2の場合を示している。なお、各符号化テーブルの信号パターンDECPATNと圧縮ビット列DECBITは、共に右側から左側に入力してくる信号を示している。この図25の場合、その信号形態は4種類となる。デコードすべきビット数はすべて2個であり、そのときの信号パターンDECPATNは「00」「10」「01」「11」の4種類となる。信号パ

ターンDECPATNが“00”のときは、2つとも優勢シンボルのため予測が当たったこととなり、フラグFAILは当たり信号の“0”となると共に、そのときの圧縮ビット列DECBITは“0”となり、ビット長LENGTHは“1”となる。

【0160】一方、信号パターンDECPATNが“10”のときは、劣勢シンボル“1”が入っており、予測が外れたこととなる。この結果、フラグFAILは、外れ信号の“1”となり、圧縮ビット列DECBITは、最初に“1”がくる。次に、“10”の前半部が“0”であるため、予測が当たり圧縮ビット列DECBITの2番目は“0”となり、“01”となる。ここで、最初に予測外れとなっているので、後半部に“1”があることとなる。このため、圧縮ビット列DECBITは、この“01”がそのまま採用される。

【0161】信号パターンDECPATNが“01”のときは、劣勢シンボル“1”が入っており、予測が外れたこととなる。この結果、フラグFAILは外れ信号の“1”となり、圧縮ビット列DECBITは最初に“1”がくる。次に、“01”の前半部が“1”であるため、予測がまた外れたこととなり、圧縮ビット列DECBITの2番目は“1”となる。信号パターンDECPATN“01”の後半部は“0”であるため、予測当たりとなり、圧縮ビット列DECBITの3番目は“0”となる。すなわち、信号パターンDECPATN“01”に対応する圧縮ビット列DECBITは、“011”となる。そして、ビット長LENGTHは“3”となる。同様にして、信号パターンDECPATN“11”に対する圧縮ビット列DECBITは、“111”となる。以上4種類の信号の対応表が図20となっている。

【0162】同様にして、テーブル番号が「2」で、runが「4」の16種類の信号の対応関係が図26に示され、テーブル番号「3」でrunが「8」の計31種類の信号の対応関係が図27に示されている。なお、図26のrunが「4」の場合では、runの値が区切りビット数pと同じとなるので、図24および図25と全く同じ関係のみのものとなるが、図27のrunが「8」の場合は、区切りビット数p（この実施の形態ではp=4）より大きくなるため、少し変更された表となる。

【0163】次に、他の表とは若干異なるこの図27の符号化テーブルの内容を説明する。この符号化テーブルでは、デコードすべき信号のビット数DECNUMは、「8」のものとして「4」のものが存在する。「8」のものは、前半部がすべて“0000”のものであり、「4」のものは、runが「8」で前半部に劣勢シンボル“1”がきた場合のものを示している。デコードすべき信号のビット数DECNUM（以下単にDECNUMとして示す）が「8」で、信号パターンDECPATN

（以下単にDECPATNとして示す）が“0000”のときは“00000000”であることを示し、予測が当たったこととなり、フラグFAIL（以下単にFAILとして示す）は当たり信号の“0”となる。そして、圧縮ビット列DECBIT（以下単にDECBITとして示す）は“0”で、ビット長LENGTH（以下単にLENGTHとして示す）は“1”となる。DECNUMが「8」で、DECPATNが“1000”のときは、“10000000”であることを示し、予測が外れたこととなり、FAILは外れ信号の“1”となる。そして、DECBITの1番目には“1”がくる。次に、前半部“0000”は予測当たりとなり、DECBITの2番目には“0”がくる。このとき、後半部“1000”に劣勢シンボル“1”が当然くることとなるため、後半部の4つの信号に対するDECBITは、特に発生しない。

【0164】後半部“1000”の中の前半部“00”は、予測当たりであり、3番目のDECBITは“0”となる。このとき、後半部“10”に劣勢シンボル“1”が当然くることとなるため、後半部の2つの信号に対するDECBITは特に発生しない。そして、この後半部“10”の前半部“0”は予測当たりとなり、4番目のDECBITは“0”となる。こうなると、最後尾に“1”があることが当然となり、特にDECBITは発生しない。よって、DECPATN“1000”に対応するDECBITは“0001”となる。そして、LENGTHは“4”となる。これが、図27のテーブル番号「3」の表の上から2番目の状態に対応する。

【0165】このような関係は、図27の符号化テーブルの第3番目から第16番目にも当てはまる。一方、図27のテーブル番号「3」の上から第17番目から第31番目までは、DECNUMが「4」となり、図21のテーブル番号「2」のものに近似する。すなわち、図21の符号化テーブルの第2番目から第16番目のものに、runが「8」として見たときの予測外れの“1”がすべて最初に付加されたものと、図27のDECNUM「4」のものは同一となる。なお、符号化テーブル3より出力される符号は、LENGTHによって指定される可変長符号になっている。

【0166】この実施の形態では、さらにrunが16のテーブルと、runが32のテーブルとを有している。その両テーブルは、図27に示すrunが8のテーブル番号「3」の考え方と同様となっており、その説明を省略する。

【0167】ストリーム生成部24は、入力の変長符号を一旦バッファリングして、出力の伝送路で定められた固定のビット幅にならして出力するものとなっている。

【0168】符号化制御部25の基本動作は、信号RUN（以下単にRUNという）によってビット列分解部2

2にビットの切り出し方法を指示し、同時に信号TABLE (以下単にTABLEという)により符号化テーブルの選択を行うものとなる。そして、符号化テーブル23からフィードバックされるFAILにより、次の符号化のためのRUNとTABLEを設定する。なお、この実施の形態では、区切りビット数pを利用した段階的な符号化を導入したため、ある予測ビット長runで符号化した際、必要に応じて途中の段階であることをこの符号化制御部25は記憶する必要がある。

【0169】この符号化制御部25の具体的な動作は、図28に示す状態遷移表に基づくものとなっている。この状態遷移表の動作について、予測当たりが続く場合を例にして説明する。ここで、初期状態は、SS1となっている。まず、状態SS1のとき、runが「1」で、TABLEは「0」である。このため、図24に示すテーブル番号「0」の符号化テーブルが使用される。そして、予測が当たった場合は、優勢シンボルが「0」が続くことであるため、入力されるビット列入力からそのDECNUMの数である「1」個分の「0」のみを符号化テーブル23に送り、テーブル番号「0」のテーブル(=図24の表)に基づいて、FAIL「0」と、DECBIT「0」と、LENGTH「1」とが出力される。そして、そのFAIL「0」が符号化制御部25に伝えられる。

【0170】符号化制御部25は、図28の状態遷移表に基づき、SS1中のFAIL「0」となるものを見つけ、次の状態として状態SS0を選択する(図28の状態遷移表の上から3番目)。このとき、信号SWは「0」となるので、シンボルの逆転はなく、そのまま「0」が優勢シンボルとなる。状態SS0においても、同様な動作の結果、状態遷移表の第1番目が選択され、状態SS3が次の状態となる。これによって、2回予測が当たったこととなる。

【0171】この状態遷移表では、2回予測が当たると、runが2倍になる。すなわち、上から7番目および8番の状態SS3となり、runが「2」となる。このように予測が当たり続けると、すなわち、入力ビット列がこの場合であると「0」であり続けると、runが「2」「2」「4」「4」「8」「8」と増えていく。また、一方、予測が外れ続けるときは、2回毎、同一runで行い小さくなっていく。すなわち、runが「8」「8」「6」「6」「4」「4」「2」「2」と小さくなっていく。そして、runが「1」のときに、予測が外れると、信号SWは反転する。

【0172】このような状態遷移表の動作のルールをまとめると、次のとおりとなる。

【0173】(1)同一の予測ビット長runでの予測が2回連続して的中したとき、予測ビット長runを2倍する。

【0174】(2)同一の予測ビット長runでの予測が

2回連続して外れたとき、予測ビット長runを1/2倍する。

【0175】(3)予測ビット長runが4以下のときは、1回で符号化を実行する。

【0176】(4)予測ビット長runが8で、DECNUM=4のときは、2回に分けて符号化を実行する。

【0177】(5)このときは、状態SS5に遷移して、予測ビット長runを「4」で、後半のビットを符号化する。

【0178】なお、信号SWの反転とは、この値が1のとき、信号SWを反転させるという意味である。

【0179】なお、図28で示す状態遷移表は、runが「8」までしか示していないが、この実施の形態では、runを最大「32」としているため、run「16」とrun「32」のものも、図示していないが同様に作成されている。また、状態遷移表としては、runが「64」以上のものにしても良い。さらに、当たりや外れが2回続いたらrunを増加させたり減少させたりするのではなく、1回毎に変えたり3回以上の数としたり、種々のパターンを採用することができ、また、このような符号化テーブルとしては、ビット数の少ないものだけを用意し、大きなビット数、例えば、16ビット以上の場合には符号化テーブルを持たないようにすることもできる。

【0180】次に、以上のような構成を有するエントロピー符号器12の動作を具体例を使用して説明する。

【0181】例えば、予測が当たり続けて、run=16となった状態で、「0000010000111100……」のような形で入力してきたビット列を符号化する場合、4ビットの区切りビット数pで区切り、まず、最初は「00000100」までを符号化することとなる。これは、劣勢シンボル「1」が第1番目の区切りビット数(=最初の4ビット)部分にはなく、第2番目(=次の4ビット)に出てくるためである。そして、次に「0011」を、そして最後に「1100」を符号化することとなる。

【0182】このため、ビット列分解部22から出力されるDECNUMは、「8」「4」「4」となる。一方、DECPATNは、「0100」「0011」「1100」(ここでは、いずれのパターンも左の数値から入力されてくるとする)となる。このような条件において、DECBITは、まず、run=16としたときの予測外れの「1」がくる。次に、「00000100」は、RUN「8」、TABLE「3」、DECNUM「8」のため、図27に示す上から5番目に相当するものであり(図27に示す各数値の場合、それぞれ右端側から入力されてくことに注意)、DECBITは「10100」となる。このため、先の「1」と合わせられた「110100」(この数値は左端から順に出力)のDECBITとLENGTH「6」が符号化テーブル部

23からストリーム生成部24に出力される。  
【0183】一方、符号化制御部25内の状態遷移表でいえば、状態SS6でDECNUM「8」のとき、FAIL「1」となったこととなり、次の状態は状態SS7となる。そして、次の「0011」は、run=8でDECNUM「4」なので、テーブル番号「3」のテーブル(=図27の符号化テーブル)が採用され、その上から19番目のものが該当し、「11011」のDECNUMとLENGTH「5」が符号テーブル3から出力される。

【0184】最後の「1100」については、前の状態が状態SS7のrun「8」、DECNUM「4」で、FAIL「1」となったため(図28の現状態コード「0111」中の1番下の状態)、状態SS5が採用される。このため、run「4」、TABLE「2」となり、図26に示すテーブル番号「2」の符号化テーブルが使用される。そして、このテーブル番号「2」のテーブルにおいて、下から5番目が該当し「11110」のDECBITと、LENGTH「5」が符号化テーブル部23からストリーム生成部24に出力される。なお、状態SS5で、FAILは「1」であるので、次は状態SS2に移る。すなわち、次の入力ビット列に対しては、run=2である図25の符号化テーブルが使用されることとなる。

【0185】以上をまとめると、入力ビット列「000010000111100」が「110100」、「11011」、「11110」の3つの圧縮ビット列として符号化されたこととなる。なお、入力ビット列や3つの圧縮ビット列は、共に先頭側から入力され、出力されていくものとする。この点、図24から図27の各符号化テーブルとは異なることに注意する必要がある。すなわち、各符号化テーブルでは、その表示の各値は、その表示の右端から順に入力し、出力するものとなっている。

【0186】そして、runは、当初「16」であったのが、この4ビットの区切りビットpで段階的に符号化していく中で、runは「2」となり、次の入力ビット列に対しては、「2」の予測ビット長runで符号化されることとなる。

【0187】一方、先に示した本実施の形態の元となる基本的プロセスで、同じ入力ビット列「000001000111100」を符号化すると、まずrun=16での予測外れの「1」、次に前半の8ビットを注目し、2番目に予測外れの「1」がきて、さらに前半の4ビット「0000」に注目し、予測当たりの「0」が3番目にくる。すると、後半部の4ビット「0100」に劣勢シンボルがくるとは確実なので、すぐに2つに分割し、前半の2ビット「01」に注目する。このため、予測外れの「1」が4番目にくる。次は、さらにこれを2分割し、前半の「0」に注目し、5番目に予測当たりの

「0」がくる。すると、後半の「1」は劣勢シンボルが確実なので、すぐに後半の2ビット「00」に注目し、予測当たりの「0」が6番目にくる。

【0188】以上の前半8ビットの符号化をまとめると、「110100」となる。これは、本実施の形態による符号化ビットと全く同じとなる。続く8ビットも同様な方法で進めていくと、これらも本実施の形態による符号化ビットと同一となる。本実施の形態の元となる基本的プロセスと本実施の形態とが異なる点は、符号化されたビット自体ではなく、1.符号化の区切り方、2.予測ビット表の変更の仕方、3.符号化テーブルの活用の3点にある。

【0189】すなわち、改良した本実施の形態では、入力ビット列に対しrunより小さい区切りビット数p(この実施の形態ではp=4)で区切り、劣勢シンボルが存在する区切り部分までで一旦符号化を区切るようにしている。先の例では、16ビットの入力ビット列が3つに区切られて符号化されている。また、本実施の形態では、次の入力ビット列に対し予測ビット長runは「2」となるのに対し、基本的プロセスの考え方では、予測外れは1回であり、runは「16」のままとする。さらに、本実施の形態の基本的プロセスの考え方では、符号化サブルーチンを再帰的に呼び出して符号化しているが、改良した本実施の形態の符号化方法では、符号化テーブル、具体的には予測ビット長run毎に符号化テーブルを用いている。

【0190】以上の3つの点は、それらが同時に利用されることによって大きな効果を生ずるが、それぞれ単独で使用されても十分効果を有する。例えば、第1の点の段階的に符号化する方法を採用すると、バッファ、例えば、ビット列分解部22やストリーム生成部24内の各バッファを小さくできるばかりか後述するマルコフモデル化によって圧縮ビット列を得ようとするときにそのバッファの容量を減少させることができる。

【0191】第2の点の予測ビット長runの変更については、入力ビット列が途中でがらっとその性質が変わるような場合に特に有効となる。先の例では、予測が当たり続けてrun=16となったのに対し、次に性質ががらっと変わったビット列、すなわち劣勢シンボルを多く含む「0000010000111100」がきたとき、改良された本実施の形態の符号化方法では、その性質に合わせ、runは「2」となり、続く入力ビット列の性質に合う確率が高いものとなり、圧縮率が高くなる。しかし、基本的プロセスで処理した場合、runは「16」のままであり、次の入力ビット列の性質にそぐわない確率の高いものとなる。なお、圧縮率の向上は、具体的には、改良前に比べ、0.5%から数%程度であるが、各プログラムソフト等が大容量化している現在では、このようなわずかな数値の向上効果も無視し得ないものとなっている。

【0192】第3の点の符号化テーブルについては、サブルーチンの再帰的呼び出しによる符号化に比べ、符号化テーブルのためのメモリ容量は若干増えるものの、符号化速度が極めて速くなる。

【0193】次に、この実施の形態のエントロピー復号器32について、図29に基づき説明する。

【0194】このエントロピー復号器32は、符号化された信号のストリームを入力するストリーム切り出し部41と、予測ビット長runに応じた複数の復号テーブルを内蔵する復号テーブル部42と、復号されたビットをストアし、所定のシンボルを出力するデコードバッファ部43と、エントロピー符号器12の状態変換部26内の状態遷移表と同じ状態遷移表を有する復号制御部44とから主に構成されている。ここで、復号制御部44には、状態遷移表を有する状態遷移部45と、マルコフモデル等により生成される復号条件を入力し、その条件毎に現在の状態の信号を状態遷移部45に与え、復号後に次の状態の信号を入力し、その復号条件の状態を記憶しておく状態記憶部46とが設けられている。

【0195】なお、デコードバッファ部43は、復号条件が入力し、その条件毎に個別に管理されるものとなっている。このため、マルコフモデルのような条件付き復号化の場合、バッファとして非常に大きなものが必要になる。しかし、本実施の形態のエントロピー復号器32では、後述するように段階的な復号を行うので、各バッファは小さいものでも十分対応でき、マルコフモデルのような条件付きの復号化でもデコードバッファ部43はそれ程大きな容量を必要としなくなる。

【0196】ストリーム切り出し部41は、復号テーブル部42から、復号したビット数を後述するLENGTHにより指示されるので、その値に基づき、復号済みビットを廃棄して、未復号ビットの先頭が、符号化されたデータとなる復号予定の符号語信号CODE（以下単にCODEという）の最下ビット（または最上位）に来るようにストリームを切り出す。なお、LENGTHを評価して、復号済みビットを廃棄するのは、デコードバッファ部43から廃棄指示DECREQ（以下単にDECREQという）があったときのみである。また、CODEは8ビット単位で送信される。

【0197】復号テーブル部42は、図30から図33に示すような各復号テーブルを内蔵し、復号制御部44が出力するテーブル番号指示信号TABLE（以下単にTABLEという）によりそれらを切り替えて使用する。そして、復号テーブル部42は、次の信号を出力する。すなわち、(1)何ビット復号したかを示す信号LENGTH（以下単にLENGTHという）で、エントロピー符号器12におけるLENGTHに相当するもの、(2)予測の当たり外れを示す信号FAIL（以下単にFAILという）で、エントロピー符号器12におけるFAILに相当するもの、(3)復号したビット・パターン

信号DECPATN（以下単にDECPATNという）で、エントロピー符号器12におけるDECPATNに相当するもの、(4)復号結果が何ビットかを示す信号DECNUM（以下単にDECNUMという）で、エントロピー符号器におけるDECNUMに相当するもの、を出力する。

【0198】図30に示すrun=1の復号テーブルは、CODEが“0”“1”の2種類に対応する各出力が記載されている。この復号テーブルは、図24のrun=1の符号化テーブルに相当するもので、符号化テーブル中のDECBITに相当するものが、この復号テーブルではCODEとなっている。図31に示すrun=2の復号テーブルは、同様に図25のrun=2の符号化テーブルに相当するものとなっている。また、図32に示すrun=4の復号テーブルでは、図26のrun=4の符号化テーブルに相当し、図33に示すrun=8の復号テーブルは、図27のrun=8の符号化テーブルに相当している。なお、各復号テーブルにおける各数値も、符号化テーブルと同様に、各数値の右端側から入力し、出力する表示となっている。また、復号テーブルとして、run=16と、run=32のテーブルも用意されている。

【0199】デコードバッファ部43は、4ビット（この実施例の場合）以下のDECPATNとDECNUMを直接的にストアし、それぞれデコードバッファ部43内のPATNREG（以下単にPATNREGという）とナンバレジスタNUMREG（以下単にNUMREGという）にストアする。そして、デコードバッファ部43の出力がqビット幅の場合、デコードバッファ部43は、1回デコード・データを出力する度にストアしたNUMREGからqを減じる。そして、NUMREGがqより小さくなら、DECREQをアクティブにして、新たなデータのデコード要求を発する。また、NUMREGが5以上のときは、信号SWで定まる優勢シンボルをデコード出力として出力する。一方、NUMREGが4以下になったら、PATNREGの値を出力する。

【0200】例えば、図33の上から5番目のCODE“00101”が復号テーブル部42に入力された場合、DECNUM=8、DECPATN=“00101”がデコードバッファ部43に入力されてくる。このとき、信号SWが“0”となっていたとし、出力を2ビット単位（これはq=2に相当）で行うとした場合、最初の2回の出力は優勢シンボルを出力すればよい。この場合SW=0なので、優勢シンボルは“0”である。したがって、“0000”を出力する。この4ビットを出力した時点で、NUMREGは4（=8-4）になっている。そこで、次のサイクルは、PATNREGの値を、順に出力する。すなわち、“0100”をこの表示の左端側から出力する。

【0201】復号制御部44の状態遷移部45は、符号



化制御部 25 の状態遷移部 26 と同じ状態遷移表を保有している。そして、状態の初期値は、SS1 であり、FAIL と DECNUM により、次の遷移先が決定され、DECREQ がアクティブのとき、その遷移先へ遷移する。

【0202】以上のように構成されるエントロピー復号器 32 は、先に示したエントロピー符号器 12 と逆のアルゴリズムによって動作する。なお、このエントロピー復号器 32 は、デコードバッファ部 43 の出力状態によって制御されるものとなっている。すなわち、デコードバッファ部 43 の NUMREG が出力ビット幅  $q$  より小さくなると、DECREQ がストリーム切り出し部 41 と復号制御部 44 へ出力される。ストリーム切り出し部 41 は、その DECREQ により復号済みビットをその LENGTH 分廃棄する。

【0203】先の例の  $run=8$  で CODE "00101" の場合、NUMREG が「8」から「4」へ、「4」から「2」、「2」から「0」へ下がる。この「2」から「0」へ下がったときに、DECREQ が発生する。そして、LENGTH が「5」であるので、CODE から復号済みの 5 ビットを廃棄する。このため、ストリーム切り出し部 41 内の CODE には、未復号ビットが最下位または最上位にきて、次の復号に備える。一方、復号制御部 44 では、 $run=8$ 、DECNUM=8 で、FAIL=1 なので、状態 SS7 へ遷移する。このため、 $run=8$  に相当する TABLE=3 を復号テーブル部 42 に向けて出力する。

【0204】この結果、復号テーブル部 42 は、図 33 のテーブル番号「3」である  $run=8$  の復号テーブルを準備する。そして、入力してきた CODE から LENGTH、DECNUM、DECPATN および FAIL が確定し、出力される。例えば、その CODE の最初が「0」であれば、CODE「0」であることが確定し、LENGTH=1、DECNUM=8、DECPATN="0000"、FAIL="0" を出力する。一方、CODE が「01011」の場合、CODE の最初が「1」であるので、まだ確定せず、次の「1」でも、3 番目の「0」でも、4 番目の「1」でも確定しない。しかし、5 番目の「0」が入った段階で「01011」であることが確定する。この確定によって、LENGTH=5、DECNUM=4、DECPATN="0100"、FAIL="1" がそれぞれ出力される。このようにして、順次、復号されていく。

【0205】このエントロピー復号器 32 は、エントロピー符号器 12 と同様に、先に述べた基本的プロセスに基づく復号に比べると、①段階的な復号によるバッファ容量の減少化②信号の性質にあった予測ビット長  $run$  の変更③復号テーブルによる復号速度の向上という各種の有利な効果を有するものとなる。

【0206】このエントロピー復号器 32 においては、

状態遷移を条件毎に個別に管理する点で、エントロピー符号器 12 と同様である。ただし、このエントロピー復号器 32 の場合は、上述したようにさらにデコードバッファ部 43 も個別に管理しなければならない。このため、デコードバッファ部 43 は、NUMREG、PATNREG に相当するレジスタを有し得る条件数分内蔵し、復号条件によって切り換えるものとなっている。

【0207】上述の説明では、分かり易さを考慮し、2 値のビット列を例にして、予測ランレングス符号化方式を説明したが、判別部 19 から出力される順位コードは多値データであり、実際は多値データを 2 値系列に変換する必要がある。例えば、ビット・プレーンに分けて、各ビット・プレーンをこの予測ランレングス符号化方式で符号化するようにしても良い。また、最上位ビットからプレーン毎にこの予測ランレングス符号化方式にて符号化を行い、「1」が出現した時点で続く下位ビットを直接ストリームに出力するようにしても良い。

【0208】この実施の形態では、この PRLC を多値系列に適用する方式として、ビット・プレーンではなくレベル・プレーンに分けて行っている。具体的には、シンボルが 8 ビットであるため、256 のレベル・プレーンに分け、その入力シンボルをグループに分け、グループ番号をこの予測ランレングス符号化方式で符号化している。すなわち、入力シンボルを図 34 に示すように、グループ分けし、まず入力シンボルがグループ番号が 0 以外かを示す判定ビットをこの PRLC で符号化する。もし入力シンボルが 0 ならこのシンボルの符号化を完了するが、そうでない場合はさらにグループ番号が 1 か 1 以外かを示す判定ビットをこの予測ランレングス符号化方式で符号化する。

【0209】このようにして、グループ番号が確定するまで、判定ビットを PRLC で符号化し、確定したグループ番号が 2 以上の場合は、該当グループにおけるシンボルを確定するため、必要とする付加ビットを直接ストリームに出力する。ただし、各グループ判定ビットは、グループ毎に独立した系列として扱い、各々個別に現状コードを管理して符号化する。この方法では、グループ番号が確定した時点で、上位の判定ビットの符号化を行わないので、処理速度が向上する。

【0210】なお、この実施の形態では、圧縮対象画素の画素インデックス 3 の数が 256 個に加え、 $(CX+1)$  が加わるため、最高 260 個が対象となり、順位としては 259 順位となる。この対策として、順位 255 をエスケープ・コードに割り当て、255 以上の順位については、このエスケープ・コードを符号化後、255 との差分を 3 ビット付加ビットとして直接圧縮ストリームに出力している。このような対策は、圧縮対象の画素インデックス 3 が 256 色をわずかに超える場合にも適用できる。

【0211】また、この実施の形態では、動画像のイン

デックスバレットは、動画ストリームの中で共通のバレットとして使用されている。さらに、このバレットは、フレーム間相関を用いた符号化では、フレーム間相関の及ぶ範囲で共通のバレットを使用している。例えば、1ピクチャーと1ピクチャーの間(GOP)毎にバレットを持つようにしたり、他の範囲で持つようにしたりしている。

【0212】また、この実施形態のエントロピー符号器12では、符号化する際、過去の符号化済み系列等により適当に条件付けを行い、符号化することで、圧縮率が向上している。すなわち、マルコフ状態信号C<sub>X</sub>により条件付けを行ってPRLC符号化を実行している。具体的には、マルコフ状態信号C<sub>X</sub>は、4状態存在し、1つの状態で8つのグループ判定ビットを個別に符号化するため、合計で32状態の現状コードを個別に管理し符号化することになる。

【0213】なお、上述の実施形態は、本発明の好適な実施形態の例であるが、これに限定されるものではなく本発明の要旨を逸脱しない範囲において、種々変形実施可能である。例えば、カラー画素データ100Aとしては、nビット(nは2以上の整数)のカラー画素データ100Aを対象とし、その符号化や復号化を行うように構成することができる。

【0214】なお、符号化や復号化を行うにあたり、図18から図22に示す改良前の方法を採用したり、図40に示す従来の算術符号型のエントロピー符号器およびエントロピー復号器を利用しても良い。また、上述の実施形態のように、プリスキャンを行うのではなく、特開平6-276041号に開示された技術を利用しても良い。すなわち、インデックスを出現頻度順に並び替えた色順位データを利用したり、最新のものを先頭へ移動する移動処理を行って得た最新出現表を利用するようにしても良い。

【0215】さらに、先頭移動処理を使用する場合、現れたカラシンボルを常に先頭へ持っていくのではなく、先頭方向へ所定の決まりにしたがって移動させるようにしても良い。このようにすると、出現確率が低いカラシンボルが出たとき、すぐに先頭へ移動しないので、出現確率の低いものに短い符号を割り当てることが無くなる。このため、出現確率が低いものが現れても符号化効率や復号化効率が悪化してしまうことはない。

【0216】なお、マクロブロックMBと小さなブロックSBによる二重判定は、マルチカラー画像以外の画像、例えば自然画の動き補償の際にも適用できる。また、動きベクトルを利用しないとしたマクロブロックMBの拾い方は、フレームのスキャンが横方向のときは縦方向に集めていき、フレームのスキャンが縦方向のときは横方向に集めていくのが好ましい。

【0217】また、動き補償を行うブロックは、一般的

に採用されている16×16ピクセルの他に8×8ピクセル、4×4ピクセル、3×3ピクセル等種々の大きさのものとすることができ、動きベクトルのためのデータ量を考慮すると4×4ピクセル以上が好ましく、効果的な動き検出を考慮すると6×6ピクセル以下が好ましい。

【0218】さらに、上述の実施形態では、符号化および復号化は、ハードとして構成されたシステムで処理しているが、すべてソフトウェアで処理するようにしても良い。また、ソフトウェアで処理する際は、その処理手順をプログラム化しCD-ROM等の記憶媒体にインストールし、その記憶媒体にパソコン等のハードに差し込んで処理したり、そのプログラムをネットワーク等を介して送受信し、パソコン等のハードディスク等へ取り込んで処理するようにしても良い。

【0219】

【発明の効果】以上説明したように、本発明のマルチカラー画像からなる動画画像の符号化装置およびその方法では、マルチカラー画像にも動き補償を適切に利用できるようになり効果的なデータ圧縮が可能となる。

【0220】また、本発明のマルチカラー画像からなる動画画像の復号化装置およびその方法では、マルチカラー画像の復号化にも動き補償を利用でき、復号化装置のメモリの低容量化が可能となると共に復号処理を高速化することができる。

【図面の簡単な説明】

【図1】本発明のマルチカラー画像からなる動画画像の符号化装置およびその方法を採用した符号化システムを示す図である。

【図2】図1の符号化システム中のインデックス変換部およびその周辺の構成を示す図である。

【図3】本発明のマルチカラー画像からなる動画画像の復号化装置およびその方法を採用した復号化システムを示す図である。

【図4】図3の復号化システム中のインデックス変換部およびその周辺の構成を示す図である。

【図5】図1の符号化システムの動き補償部の動作結果を説明するための図で、(A)は各ピクチャー(フレーム)の種類と入力順序を示す図で、(B)はそれのときの符号化順序を示す図である。

【図6】図1の符号化システムおよび図3の復号化システムで採用されるマクロブロックと小さなブロックおよびその関係を示す図である。

【図7】図1および図3の符号化システムおよび復号化システムにマルコフモデルとして採用される参照画像の配置を説明するための図で、(A)は通常の参照画像を、(B)は先頭ラスタの場合を、(C)は先頭画像の場合をそれぞれ示している図である。

【図8】図1および図3の符号化システムおよび復号化システムに採用される参照画像の状態と、状態信号ST

と、色数と、マルチカラー状態信号 C X との関係を示す図である。

【図 9】図 1 の符号化システムに採用される変換テーブル生成部の参照順位テーブル作成の機能を説明するための図で、(A) は符号化対象画素と参照画素が一致したときの度数の例を示す図で、(B) (C) はそれぞれの度数が所定の関係となったときの参照画素の順位を示す図である。

【図 10】図 1 および図 3 の符号化システムおよび復号化システムに採用される参照順位テーブルの例を示す図 10 である。

【図 11】図 1 の符号化システムに採用される変換テーブル生成部の画素インデックス変換テーブル作成機能を説明するための図で、入力してきた画素インデックスに対応する画素が参照画素中に無かった度数を示す図である。

【図 12】図 1 の符号化システムに採用される変換テーブル生成部の画素インデックス変換テーブル作成機能を説明するための図で、入力してきた画素インデックスに対応する画素が参照画素中に無かった度数の順位を示す 20 図である。

【図 13】図 1 の符号化システムにおける入力してきたカラーパレットから変換カラーパレットへの変換を説明するための図である。

【図 14】図 1 の符号化システムにおけるブリスキャン時の動作を示すフローチャートである。

【図 15】図 1 の符号化システムにおける符号化プロセス時の動作を示すフローチャートである。

【図 16】図 1 の符号化システムにおいて動き補償を行わないと判断されたマクロブロックの符号化方法を説明するための図で、(A) は動き補償を行わないマクロブロックの位置の例を示し、(B) はその一部拡大図である。

【図 17】図 3 の復号化システムにおける復号化プロセスを示すフローチャートである。

【図 18】本発明で採用するエントロピー符号器およびエントロピー復号器で使用しているアルゴリズムの概要を説明するための図で、注目系列と予測ビット数 run との関係を示す図である。

【図 19】図 18 の注目系列を分割した状態を示す図 40 である。

【図 20】図 19 の前半部注目系列をさらに分割した状態を示す図である。

【図 21】図 18 から図 20 に示すエントロピー符号器内の符号化プロセスを説明するためのフローチャートで、符号化メインルーチンを示すフローチャートである。

【図 22】図 18 から図 20 に示すエントロピー符号器内の符号化プロセスを説明するためのフローチャートで、符号化サブルーチンを示すフローチャートである。 50

【図 23】本発明の実施の形態で採用するエントロピー符号器の構成を示すブロック図である。

【図 24】図 23 のエントロピー符号器の符号化テーブル内の符号化テーブルを示す図で、予測ビット長が「1」の場合のテーブルを示す図である。

【図 25】図 23 のエントロピー符号器の符号化テーブル内の符号化テーブルを示す図で、予測ビット長が「2」の場合のテーブルを示す図である。

【図 26】図 23 のエントロピー符号器の符号化テーブル内の符号化テーブルを示す図で、予測ビット長が「4」の場合のテーブルを示す図である。

【図 27】図 23 のエントロピー符号器の符号化テーブル内の符号化テーブルを示す図で、予測ビット長が「8」の場合のテーブルを示す図である。

【図 28】図 23 のエントロピー符号器の状態遷移部内の状態遷移表を示す図である。

【図 29】本発明の実施の形態で採用するエントロピー復号器の構成を示すブロック図である。

【図 30】図 29 のエントロピー復号器の復号テーブル内の復号テーブルを示す図で、予測ビット長が「1」の場合のテーブルを示す図である。

【図 31】図 29 のエントロピー復号器の復号テーブル内の復号テーブルを示す図で、予測ビット長が「2」の場合のテーブルを示す図である。

【図 32】図 29 のエントロピー復号器の復号テーブル内の復号テーブルを示す図で、予測ビット長が「4」の場合のテーブルを示す図である。

【図 33】図 29 のエントロピー復号器の復号テーブル内の復号テーブルを示す図で、予測ビット長が「8」の場合のテーブルを示す図である。

【図 34】本発明で採用するアルゴリズムを本実施の形態で示す多値系列データに適用した場合の例を説明するための図で、入力シンボルを複数のグループに分けた状態を示す図である。

【図 35】従来のマルチカラー画像の符号化システムおよび復号化システムのブロック図である。

【図 36】従来の符号化対象画素データに対する参照画素データの説明図である。

【図 37】従来のパラメータテーブルを示す図である。

【図 38】状態縮退器を有する従来のマルチカラー画像の符号化システムおよび復号化システムのブロック図である。

【図 39】従来の縮退テーブルの一例を示す図である。

【図 40】従来の算術符号型のエントロピー符号器およびエントロピー復号器の説明図である。

【図 41】従来の本発明で使用するマルチカラー画像のインデックスを説明するための図である。

【図 42】従来の合成された色順位テーブルの作成原理を示す説明図で、(A) は各画素の配置関係を示し、(B) は各画素のカラーシンボルを示し、(C) は合成

された色順位テーブル（最新出現表）を示す図である。

【符号の説明】

1 符号化システム

2 動き予測部

2a 判断手段

3 復号化システム

4 動き補償部

4a 判断手段

7、8 インデックス変換部

10 ラインバッファ（周辺画素生成手段）

11 マルコフモデル作成部（縮退手段）

\* 12、12a エントロピー符号器（エントロピー符号化手段）

20 合成部

30 ラインバッファ

31 マルコフモデル生成部（縮退手段）

32、32a エントロピー復号器（エントロピー復号化手段）

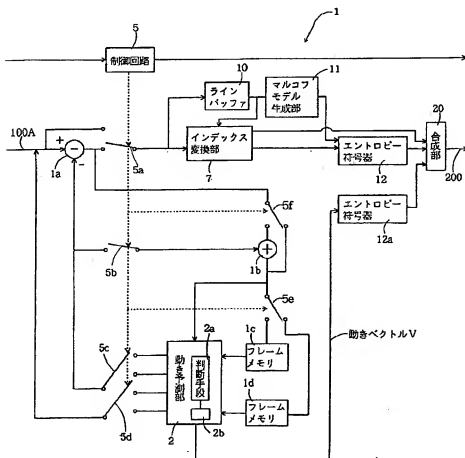
33 分離部

100A 符号化されるカラー画素データ

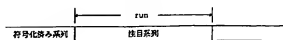
10 100B 復号化されたカラー画素データ

\* 200 符号化データ

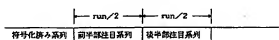
【図1】



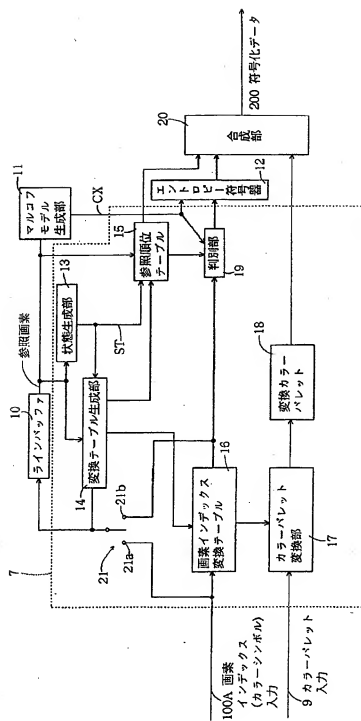
【図18】



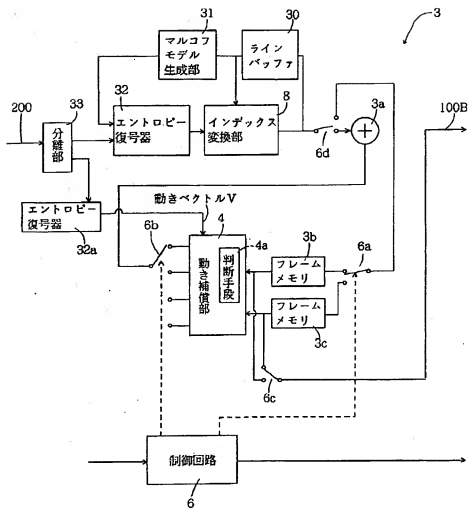
【図19】



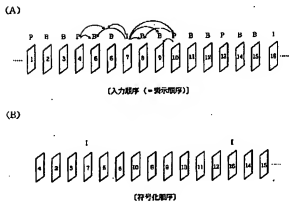
【図2】



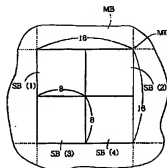
【圖 3】



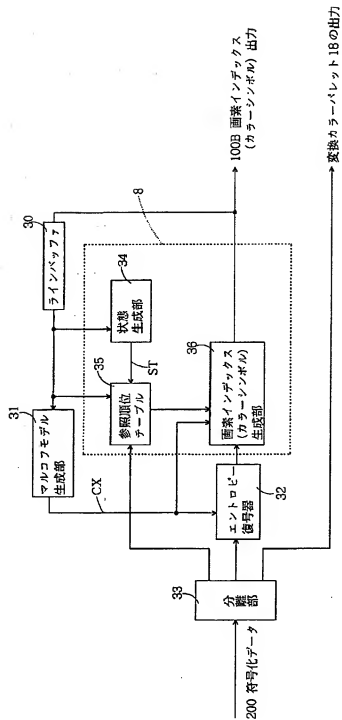
【图 5】



【图6】



【図4】



【図 7】

(A)

R [3]	R [1]	R [2]
R [0]	buf [1]	

通常の参照演算

(B)

R [3] = 0	R [1] = 0	R [2] = 0
R [0]	buf [1]	

先頭ラスタ

(C)

R [3] = 0	R [1] = 0	R [2] = 0
R [0] = 0	buf [1]	

先頭演算

【図 11】

前演インデックス (カラーシンボル)	参照演算中になかった度数
C <sub>0</sub>	N <sub>0</sub>
C <sub>1</sub>	N <sub>1</sub>
C <sub>2</sub>	N <sub>2</sub>
⋮	⋮
C <sub>n</sub>	N <sub>n</sub>
⋮	⋮
C <sub>255</sub>	N <sub>255</sub>

【図 8】

参照演算の式	ST	色数	EX (マクロ 使用数)
R [0] = R [1] - R [2] = R [3]	0	1	0
R [0] = R [1] = R [2] = R [3]	1	2	1
R [0] = R [1] - R [2] = R [3]	3	2	1
R [0] = R [2] = R [3] = R [1]	6	2	1
R [1] = R [2] = R [3] = R [0]	7	2	1
R [0] = R [1] = R [2] = R [3]	9	2	1
R [0] = R [2] = R [1] = R [3]	11	2	1
R [0] = R [3] = R [1] = R [2]	13	2	1
R [0] = R [1] R [0] = R [2] R [0] = R [3] R [0] = R [3]	16	3	2
R [0] = R [2] R [0] = R [1] R [0] = R [3] R [1] = R [3]	18	3	2
R [0] = R [3] R [0] = R [1] R [0] = R [2] R [1] = R [3]	21	3	2
R [1] = R [2] R [0] = R [1] R [1] = R [3] R [0] = R [3]	24	3	2
R [1] = R [3] R [0] = R [1] R [1] = R [2] R [0] = R [2]	27	3	2
R [2] = R [3] R [0] = R [1] R [1] = R [2] R [0] = R [2]	30	3	2
R [0] = R [1] R [0] = R [2] R [0] = R [3] R [2] = R [3]	33	4	3
R [1] = R [2] R [0] = R [3]			

【図 9】

(A)

参照演算	符号化対象演算と一致した数
R [0] (= R [1] = R [3])	NA
R [2]	NB

(B)

NA ≥ NB のとき

参照演算位置	参照演算
0 位	R [0]
1 位	R [2]

(C)

NA &lt; NB のとき

参照演算位置	参照演算
0 位	R [2]
1 位	R [0]



【図10】

参照要素位置				
ST	0位	1位	2位	3位
0	0			
1	0	3		
3	2	0		
5	1	0		
7	0	1		
9	0	2		
11	1	0		
13	0	1		
15	0	2	3	
18	1	0	3	
21	2	0	1	
24	1	0	3	
27	0	2	1	
30	1	2	0	
33	2	3	0	1

【図12】

画素インデックス (カラーシンボル)	項位
C <sub>0</sub>	1
C <sub>1</sub>	⋮
C <sub>2</sub>	0
⋮	⋮
C <sub>n</sub>	255
⋮	⋮
C <sub>255</sub>	⋮

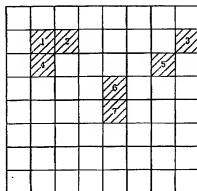
【図13】

入力された カラーパレット		→	変換された カラーパレット	
C <sub>0</sub>	RGB <sub>0</sub>		C <sub>0</sub>	RGB <sub>2</sub>
C <sub>1</sub>	RGB <sub>1</sub>		C <sub>1</sub>	RGB <sub>0</sub>
C <sub>2</sub>	RGB <sub>2</sub>		⋮	⋮
⋮	⋮		⋮	⋮
C <sub>n</sub>	RGB <sub>n</sub>		⋮	⋮
⋮	⋮		⋮	⋮
C <sub>255</sub>	RGB <sub>255</sub>		C <sub>255</sub>	RGB <sub>n</sub>

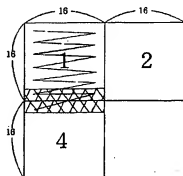
$(C_2 \rightarrow C_0)$   
 $(C_0 \rightarrow C_1)$   
 $(C_n \rightarrow C_{255})$

【図16】

(A)



(B)



【図24】

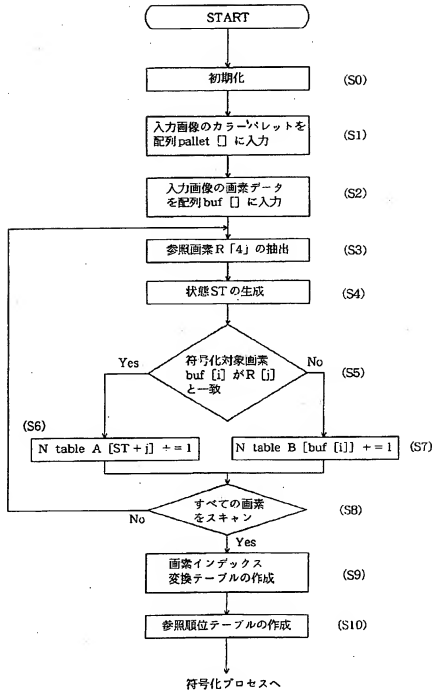
入力		出力		
DECNUM	DECPATN	DECBIT	LENGTH	FAIL
1	***0	*****0	1	0
1	***1	*****1	1	1

Table No.0 (run = 1)

【図20】

	run/4	run/4	
符号化済み系列	前半部系列	後半部系列	

【図14】

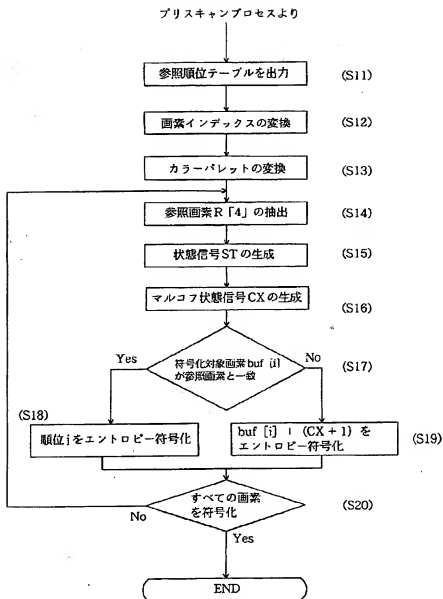


【図30】

入力		出力			
CODE	LENGTH	DEGNUM	DECPATN	PAR	
*****0	1	1	****0	0	
*****1	1	1	***1	1	

Table No.0 (run = 1)

【図 15】



【図 39】

(A)

C	B
A	X

X : 符号化対象画像  
A,B,C : 参照画像

(B)

条 件	Sx
A = B = C	S1
A = B ≠ C	S2
A = C ≠ B	S3
A ≠ B = C	S4
A ≠ B ≠ C	S5

【図 25】

入力		出力		
DECRUM	DECPATN	DECRIT	LENGTH	FAIL
2	**00	*****0	1	0
2	**10	*****01	2	1
2	**01	*****011	3	1
2	**11	*****111	3	1

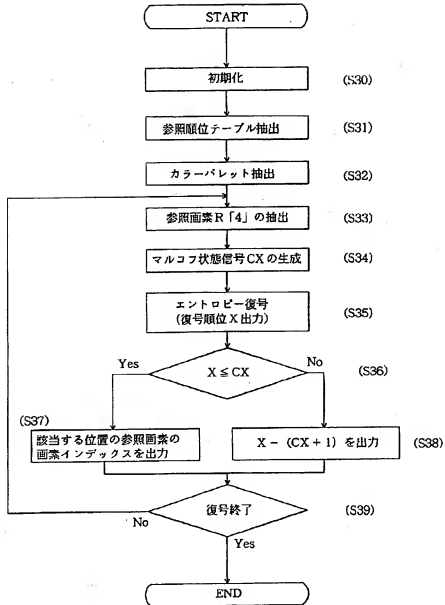
Table No.1 (run = 2)

【図 31】

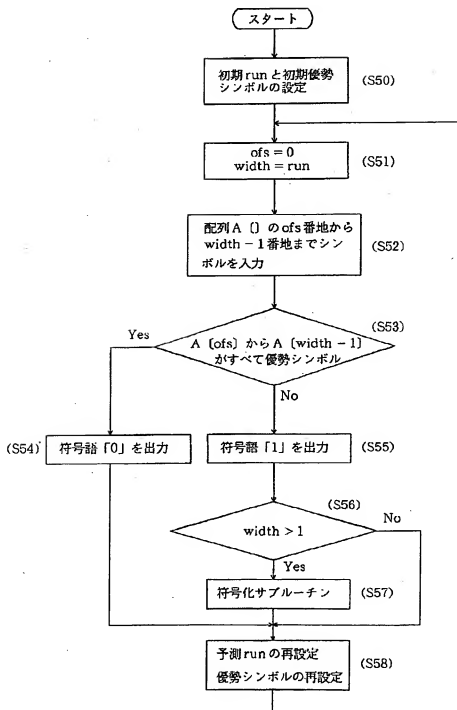
入力		出力			
CODE	LENGTH	DECRUM	DECPATN	FAIL	
*****0	1	2	**00	0	
*****01	2	2	**10	1	
*****011	3	2	**01	1	
*****111	3	2	**11	1	

Table No.1 (run = 2)

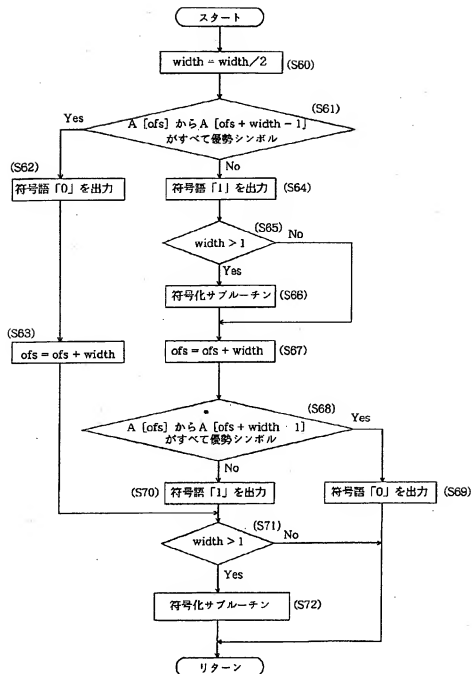
【図17】



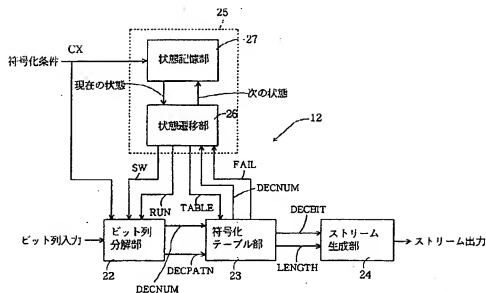
【図21】



【図22】



【図23】



【図26】

入力		出力			
DECNUM	DECPATN	DECBIT	LENGTH	FAIL	
4	0000	*****0	1	0	
4	1000	*****001	3	1	
4	0100	****0101	4	1	
4	1100	****1101	4	1	
4	0010	*****011	4	1	
4	1010	***01011	5	1	
4	0110	**011011	5	1	
4	1110	**111011	6	1	
4	0001	***00111	6	1	
4	1001	**010111	6	1	
4	0101	*0110111	7	1	
4	1101	*1110111	7	1	
4	0011	**001111	5	1	
4	1011	**011111	6	1	
4	0111	*0111111	7	1	
4	1111	*1111111	7	1	

Table No.2 (run = 4)

【図27】

入力		出力			
DECNUM	DECPATN	DECBIT	LENGTH	FAIL	
8	0000	*****0	1	0	
8	1000	*****001	4	1	
8	0100	***01001	5	1	
8	1100	***11001	5	1	
8	0010	***00101	5	1	
8	1010	**010101	7	1	
8	0110	**110101	7	1	
8	1110	*1110101	7	1	
8	0001	**001101	6	1	
8	1001	*0101101	7	1	
8	0101	01101101	8	1	
8	1101	11101101	8	1	
8	0011	**011101	6	1	
8	1011	*0101101	7	1	
8	0111	*0111101	7	1	
8	1111	11111101	8	1	
4	1000	*****011	4	1	
4	0100	***0111	5	1	
4	1100	***11011	5	1	
4	0010	***00111	5	1	
4	1010	**010111	6	1	
4	0110	*0110111	7	1	
4	1110	*1110111	7	1	
4	0001	**001111	6	1	
4	1001	*0101111	7	1	
4	0101	01101111	8	1	
4	1101	11101111	8	1	
4	0011	**011111	6	1	
4	1011	*0111111	7	1	
4	0111	01111111	8	1	
4	1111	11111111	8	1	

Table No.3 (run = 8)

【図28】

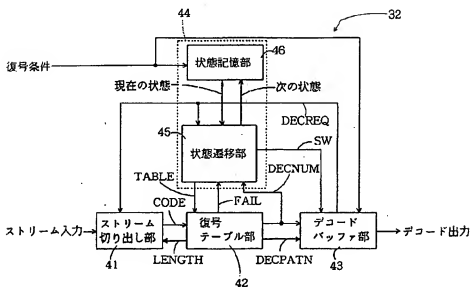
現状態	復条件であり、また 次状態への出力		入力してくる 結果		次状態番号		
現状態コード	現在の状態 run ORUN	Table (TABLE)	FAIL	DECNUM	次の状態	SWの脱出	
0000	SS0	1	0	0	1	SS3	0
	SS0	1	0	1	1	SS1	1
	SS1	1	0	0	1	SS0	0
0001	SS1	1	0	1	1	SS1	1
	SS2	2	1	0	2	SS0	0
	SS2	2	1	1	2	SS3	0
0010	SS3	2	1	0	2	SS2	0
	SS3	2	1	1	2	SS0	0
	SS4	4	2	0	4	SS7	0
0100	SS4	4	2	1	4	SS6	0
	SS5	4	2	0	4	SS4	0
	SS6	4	2	1	4	SS2	0
0101	SS6	8	3	0	8	SS6	0
	SS6	8	3	1	8	SS7	0
	SS6	8	3	0	8	SS6	0
0110	SS7	8	3	1	8	SS4	0
	SS7	8	3	1	4	SS6	0
	SS7	8	3	1	4	SS6	0
1000	...	...	...	...	...	...	...
1001	...	...	...	...	...	...	...
1010	...	...	...	...	...	...	...
1011	...	...	...	...	...	...	...

【図32】

入力 CODE	LENGTH	DECNUM	DECPATN	FAIL
*****0	1	4	0000	0
*****601	3	4	1000	1
*****0101	4	4	0100	1
*****1101	4	4	1100	1
*****0011	4	4	0010	1
*****1011	5	4	1010	1
**011011	6	4	0110	1
**111011	6	4	1110	1
***00111	6	4	0001	1
**010111	6	4	1001	1
*0110111	7	4	0101	1
*1110111	7	4	1101	1
***01111	6	4	0011	1
**011111	6	4	1011	1
*0111111	7	4	0111	1
*1111111	7	4	1111	1

Table No.2 (run = 4)

【図29】





【図 33】

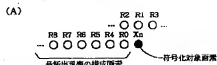
入力 CODE	出力 LENGTH	DECNUM	DECPATN	FAIL
*****0	1	8	0000	0
***00001	4	8	1000	1
**01001	5	8	0100	1
**11001	5	8	1100	1
**00101	5	8	0010	1
**010101	6	8	0101	1
*0110101	7	8	0110	1
*1110101	7	8	1110	1
**001101	6	8	0001	1
*0101101	7	8	1001	1
01101101	8	8	0101	1
11101101	8	8	1101	1
**0011101	8	8	0011	1
*0111101	8	8	1011	1
01111101	8	8	0111	1
11111101	8	8	1111	1
****0011	4	4	1000	1
***0111	5	4	0100	1
**011011	5	4	1100	1
**001111	5	4	0010	1
**010111	6	4	1010	1
*0110111	7	4	0110	1
*1110111	7	4	1110	1
**001111	6	4	0001	1
*0101111	7	4	1001	1
01101111	8	4	0101	1
11101111	8	4	1101	1
**011111	6	4	0011	1
*0111111	7	4	1011	1
01111111	8	4	0111	1
11111111	8	4	1111	1

Table No.3 (run = 8)

【図 34】

入力シンボル	グループ番号	付加ビット数
0	0	0
1	1	0
2,3	2	1
4,5,6,7	3	2
8~15	4	3
16~31	5	4
32~63	6	5
64~127	7	6
128~255	8	7

【図 42】



画素優先順位

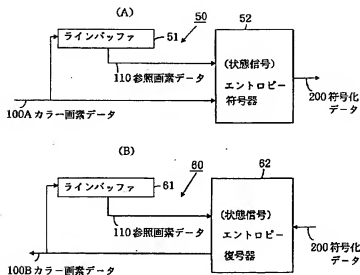
学習なし: R0, R1, R2, ..., R8... (全画素固定)

学習適応: R0, R1, R2, R3, R4, R5, R6, R7, R8, R9, R10, R11, R12, R13, R14, R15, R16, R17, R18, R19, R20, R21, R22, R23, R24, R25, R26, R27, R28, R29, R30, R31, R32, R33, R34, R35, R36, R37, R38, R39, R40, R41, R42, R43, R44, R45, R46, R47, R48, R49, R50, R51, R52, R53, R54, R55, R56, R57, R58, R59, R60, R61, R62, R63, R64, R65, R66, R67, R68, R69, R70, R71, R72, R73, R74, R75, R76, R77, R78, R79, R80, R81, R82, R83, R84, R85, R86, R87, R88, R89, R90, R91, R92, R93, R94, R95, R96, R97, R98, R99, R100, R101, R102, R103, R104, R105, R106, R107, R108, R109, R110, R111, R112, R113, R114, R115, R116, R117, R118, R119, R120, R121, R122, R123, R124, R125, R126, R127, R128, R129, R130, R131, R132, R133, R134, R135, R136, R137, R138, R139, R140, R141, R142, R143, R144, R145, R146, R147, R148, R149, R150, R151, R152, R153, R154, R155, R156, R157, R158, R159, R160, R161, R162, R163, R164, R165, R166, R167, R168, R169, R170, R171, R172, R173, R174, R175, R176, R177, R178, R179, R180, R181, R182, R183, R184, R185, R186, R187, R188, R189, R190, R191, R192, R193, R194, R195, R196, R197, R198, R199, R200, R201, R202, R203, R204, R205, R206, R207, R208, R209, R210, R211, R212, R213, R214, R215, R216, R217, R218, R219, R220, R221, R222, R223, R224, R225, R226, R227, R228, R229, R230, R231, R232, R233, R234, R235, R236, R237, R238, R239, R240, R241, R242, R243, R244, R245, R246, R247, R248, R249, R250, R251, R252, R253, R254, R255.

学習により可変

固定

【図 35】



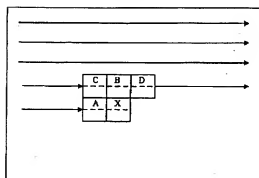
(B)



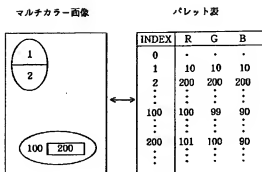
最新出現画素

色順位	カラーシンボル
1位	C4
2位	C3
3位	C8
4位	C5
5位	C2
6位	CD
7位	C8
...	...
16位	Cn

【図36】



【図41】

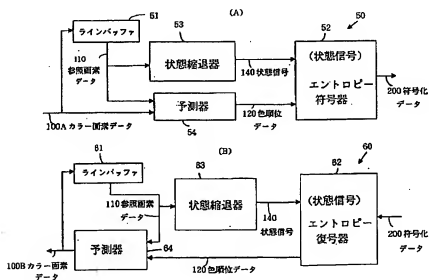


【図37】

インデックス		各カラーシンボルの出現頻度					P
		A	B	C	D	E	
参照画像 パターン	0	64,936	15,882	26,320	7,669	5,634	2,634
	1	5,553	38,214	18,776	20,385	6,046	963
	2	33,365	18,247	22,563	4,423	10,012	1,369
	3	9,569	33	13,456	582	356	224
	4	11,236	36,877	26,650	1,096	532	1,096
		65635	4,963	15,689	45,878	2,236	3,326
							63,697

A,B,C ..... P: カラーシンボル

【図38】



【図40】

